The Theories of Design Patterns

and their

Practical Implications

exemplified for

E-Learning Patterns

Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Geographischen Fakultät
der Katholischen Universität Eichstätt-Ingolstadt
vorgelegt von

Christian Kohls

Eichstätt, 2013

Referent:	Prof. Dr. Andreas Harrer
Korreferent:	Prof. Dr. Symeon Retalis
Tag der Disputation:	20. Januar 2014

The Theories of Design Patterns

and their

Practical Implications

exemplified for

E-Learning Patterns

Christian Kohls

Foreword

Patterns have changed my life…This is how I started a presentation at the annual meeting of the "Gesellschaft für Medien in der Wissenschaft" (GMW) in 2008. This statement is true to the present day. Patterns have changed my way of looking at the world.

In the beginning, patterns have fascinated me from a most practical point of view. When I started studying computer science I tried to develop a multimedia authoring tool. It was a failure – the architecture soon had a complexity that could not be handled any more. It was frustrating. Then there was a course on software design and the recommended reading was "Design Patterns – Elements of Reusable Object-Oriented Software". I felt more than excited – an excitement that is shared by many software developers as I know now. Not only did the patterns solve some of my most wicked problems I had in my design. They made me aware of problems that I had overseen before. Accordingly I had never even started to address them. The patterns helped me to build on the knowledge of experienced designers. A new architecture of the authoring tool was based on appropriate patterns and the software became solid and maintainable. My fascination was all practical and pragmatic. I did not reflect on patterns on a meta-level then.

Never had I thought that patterns would lead me to a quite theoretical exploration as it can be found in the present work. In this thesis I do not only approach patterns from a scientifically perspective. The theoretical framework developed in this work theorizes about the theoretical character of patterns. At this point I deliberately use the term "theory" three times in one sentence. As every epistemological approach it consists of meta-theoretical reasoning. You cannot be more far away from the practical. But practice and theory are much more intertwined than we often realize. The very notion of a pattern means that there is recurrence between structural arrangements. But such invariance can only be extracted if we abstract from some features. In other words: when we get theoretical.

It is a curiosity that only at the end of my work I realized that the subject of my thesis could as well have been abstraction. Abstraction is implicitly discussed in all of the chapters. But the explication of different ways of abstractions was added very late in the process of writing. I was surprised how little systematic papers on abstraction exist. The discussion about abstraction is included in many resources about science; but it is rarely a subject of its own. According to services such as Google Scholar, WorldCat or Amazon most publications about abstractions are from the following domains: art, computer science, psychology and philosophy. There are now more computer science books on abstraction available than there are on abstract art or the art of abstraction. There are many different styles of abstraction and patterns are a specific way to abstract without getting too abstract.

A pattern is at the same time abstract and concrete, theoretical and practical, liberating and constraining, particular and universal, analytical and whole. To make these statements no longer contradictions is one of the deeper motivations of this work.

From a more down-to-earth perspective the goal of this thesis is to integrate the abundance of different understandings of what a pattern is and derive practical guidance to find and document valuable patterns in an accessible way. After participating in more than 70 Writer's Workshop sessions and being the "shepherd" about 15 times I feel the need to justify some of my frequent recommendations more solidly.

Acknowledgements

The writing of this thesis has been a long path and there are many people who shaped it significantly. When I decided to make a big change of direction from pure psychological questions towards information and computer science, it was Prof. Dr. Andreas Harrer who instantly agreed to supervise me and I am very thankful for his support, trust, confidence and encouragement. I am also very grateful to my second reviewer Prof. Dr. Symeon Retalis who has been a friend and member of the pattern community for a long time. Prof. Dr. Stephan Schwan supported my research in the beginning when the research questions have been more psychological. Prof. Dr. Uwe Schmidt once supervised both my diploma and master thesis and he was the first who introduced me to design patterns and pointed out that the authoring tool I developed contained many patterns as well. The patterns of (educational) interactive graphics have been mined together with Tobias Windbrake who is always the best to exchange thoughts with. Dr. Stefanie Panke questioned my faithful believe in the "truth" of patterns. First I defended the reality of patterns by referring to schemas that are commonly accepted. Then I had to accept that it is exactly schema theory that tells us that perceived patterns are constructed. Stefanie has also helped me a great deal to get the philosophical and epistemological questions right (while I still manage to get some very wrong). Prof. Dr. Katharina Scheiter encouraged and supported me to review the similarities between patterns, schema theory and problem schemas in particular. Dr. Joachim Wedekind was a supporter of my pattern enthusiasm from the very beginning when I started to work at the Knowledge Media Research Center (KMRC) in Tübingen and provided me a lot of advice and experience on pedagogical questions; together we edited a book about e-learning patterns. Dr. Birgit Gaiser and Simone Haug provided very constructive critique to the pattern approach and highlighted the similarities to established qualitative research methods. The great discussions we had in the e-teaching.org team drove many insights I had on patterns. Several new articles at e-teaching.org are now written following the pattern format. When Dr. Anne Thillosen started as a new project leader of e-teaching.org she found already a small group of pattern enthusiasts and she was very supportive in pushing the idea forward. Throughout my time at the Knowledge Media Research Centre I enjoyed many discussions, encouragements and great feedback about patterns; I would like to use this opportunity to thank Prof. Dr. Dr. Friedrich W. Hesse (founding director), Prof. Dr. Ulrike Cress (head of my work group, the knowledge construction lab), the e-teaching.org team and all my colleagues at the KMRC for their inspirations. Many suggestions and ideas also came from the GMW community and we had several discussions about patterns in the blogosphere. Thanks to Prof. Dr. Gabi Reinmann who triggered my reflections about what is new about the pattern approach and gave me the opportunity to present a status of my thesis in her doctorial colloquium as a guest. Dr. Frank Vohle and I had several discussions about Blohm's term of "wholeness" and its meaning for education. I am especially thankful to Univ.-Prof. Dr. Peter Baumgartner. We had long discussions about Alexander's "Nature of Order", the "fifteen fundamental properties", the objectivity of "wholeness" and educational patterns on several occasions, including a blog discussion that fills over 80 letter-size pages. Peter pushed me to investigate the latest work of Alexander and to critically reflect upon the current practice of pattern authors. This push is mirrored in many parts of this thesis and expanded my path when I thought I was already near to the end.

There is one large group of people that need an extraordinary thanks and a big hug: the pattern community that meets at various *PLoP conferences around the world. There is Linda Rising who first had a look at my path metaphor and encouraged me to follow-up on that thought. There are my shepherds who provided outstanding feedback to my papers: Ralph Johnson, Dave West, Andreas Rüping, Linda again, Fujino Terunob, Tim Wellhausen, Nuno Flores, Rebecca Wirfs-Brock, Dietmar Schütz, Jenny Quillien, Andreas Fiesser and Ademar Aguiar. More than 100 people provided feedback in Writer's Workshops to parts of this thesis as I have submitted some chapters to PLoP conferences. It is impossible to name all the people I have shared great moments with at those conferences. But I want to list some special people from the pattern community who have influenced this work and my thinking: Allan Kelly, Axel Schmolitzky, Bob Hanmer, Christian Köppe, Claudius Link, George Platts, James Noble, Joseph Yoder, Klaus Marquardt, Lise Hvatum, Michael Weiss, Neil Harrison, Paris Avgeriou, Reinhard Bauer, Richard Gabriel, Takashi Iba, Till Schümmer, Uwe Zdun, Yishay Mor. Group Sneeze!

Last but not least I want to thank my family and friends in Kiel, Hamburg, Berlin, Tübingen, and Chapel Hill for their patience. Guys, I hope to see you more frequently in the future!

Table of Content

Figures

Tables

1 Introduction

1.1 Types of patterns

The world we live in is made of patterns. Every meaningful recurrent structure is a pattern: the cycles of the stars, the changes in weather, the leaves on the trees, animal species, tapestry on the wall, getting to work every morning, people falling in love, people going to war, business operations, organizational structures, behaviours of individuals, the soda bottles leaving the factory.

Every theory and every law describes a regular pattern. Every rule of thumb and every workflow is a pattern. Every template and every stencil can create patterns. The letters of the alphabet are patterns and so are words and styles of writing.

Without patterns we would fail to communicate and recognize objects. In fact, we would not be able to learn anything. Everything that we "know" is some invariant in the world, something that recurs – a pattern.

The range of different kinds of patterns, as we have seen in the given examples, is enormous. There are variations in the scale of the pattern (i.e. the size of the recurrent structure – its granularity), the level of abstraction and the order of regularity: two soda bottles are almost identical; two work days can be very different. Some patterns let us predict the future; other patterns let us operate in a regular way; some patterns can be identified in the creation of artefacts.

1.1.1 Design patterns

The focus of this thesis will be on patterns that we can design – design patterns. We will use a broad interpretation of the term "design". Whenever an intervention of human beings is possible, we shall speak of designable patterns. We can designs towns, buildings, rooms; we can design software, user interfaces, and interactions; we can design a TV, a car, a dishwasher, or a mobile phone; we can design educational systems, lessons, resources, assessments, learning software; we can design organizational structures, methodologies, and business models; we can design our work day, the plan for our holidays, the arrangement on the breakfast table and the next steps in our life.

That we can design something does not mean that we are totally free of choice: the family budget constraints the next holidays, a person's education constraints the choice of workplace, cognitive capabilities and talent constrain what we can learn, administrative policies constrain which learning opportunities are offered, physical laws constrain what we can engineer, programming languages and computing power constrain which software we can develop, natural resources constrain which cities and houses we can build.

In spite of the constraints, there are still infinite numbers of design options to choose from. Design means that we are capable to shape the forms within the given constraints. Contrast this to patterns of nature: we cannot shape the paths of the stars, we cannot shape the weather, we cannot shape any physical law, and we cannot shape the processes of biology. Although it is true that modern man has started to interfere the later: creating environments to let plants grow in a specific way (agriculture or gardens), chemical pills to alter internal processes in the body (pharmacy and medicine), and artificial selection of new species by rearing and now gene-manipulation. The degree of possible intervention varies in different domains; the predictability of the outcome of an intervention is also of different degrees. For example, we can statistically measure how many times a medicine cures an ill person. But it is harder to predict whether a specific development process (such as SCRUM) or a learning setup (such as a group meeting) will have the hoped effects.

1.1.2 Emerging patterns

Not every pattern that is designable was consciously designed. During the industrial revolution, patterns emerged in the neighbourhoods of London without deliberate design (Johnson, 2002). Many inventions have been found by accident and moments of "Eureka!" are never planned. The fact that these patterns have not been designed originally does not mean that they cannot be designed deliberately in the future. For example, if you put a chair in your room to a different location because you are cleaning or searching for a dropped coin and you accidently discover that this location fits much better one can hardly speak of a design process. However, to let the chair on its new position is a conscious decision. It becomes a design by the fact that it is an artificial set-up; there is no natural law to leave the chair on its new position, one can change it at any time. Moreover, one can repeat the accidently found set-up in other rooms. Good design potentially emerges from conscious planning, accidental variations, and the forces of nature. Constant evaluation of the appropriateness of a local situation can lead to good designs without central planning.

The goal of design patterns is to capture proven designs or “best” practices. That is they suggest interventions that have shown to have positive effects. It does not matter whether the original design emerged by planning or naturally. The important fact is that we can create new instances of the good design based on observations what has worked in the past. Those new instances are created by deliberate intervention.

1.1.3 Patterns to improve our environments

A conscious choice for good design leads to improvements of our environments. In Christopher Alexander’s view when a place becomes more alive, more whole and has a “quality without a name” its goodness cannot be reduced to single properties – it has at the same time freedom, life, wholeness, comfort, and harmony (Alexander, 1979). This quality is found in designs that have an inner balance and that do not contradict their environmental needs.

Such improvements occur if we replace a bad design by a better form. They also occur when a new system is implemented based on good designs, rather than problematic or untested solutions. Sometimes an intervention cannot guarantee an improvement but makes it more likely.

In order to accept a pattern and recognize it as good design, we need to understand the problems it solves. By understanding the problem we are not only justifying a pattern; we also become aware of the problems that current designs or practices may cause for us. Very often we feel uncomfortable at a place or in a situation without realizing what the problem is. Design patterns do not only provide good solutions to common problems – they shed light on hidden problems (Coplien, 1998e).

1.1.4 Solutions in a context

No intervention is good (or bad) by itself but only by the effect it produces – the quality that emerges. However, the effect does depend on both the intervention and the situation or context it is applied in. The same medicine can have good or bad effects depending on the state of the patient. The increase or decrease of taxes will cause different effects depending on the economic situation and its current rate.

Well meant interventions to help developing countries have had often disastrous effects because the local situation was not taken into account. The same problem needs to be tackled different in various situations because its causes are different. The pattern approach analyses the deeper structure of a problem in terms of forces that provide a rationale for the form of the solution.

A form only becomes a solution when it is applied in the appropriate context. It is meaningless to speak of solutions or good designs without considering the context. Explicitly discussing the context to which a form fits allows the adequate usage of methods, tools and technologies. Such literacy in selecting the right tools for the task at hand is very well understood by every craftsman. The hammer cannot be used to saw (wrong context of use) but that does not lower its value to hit a nail (correct context of use). This works the other way around as well. Hitting a nail into the wall can be supported by a hammer rather than a saw or a newly invented tool altogether. Moreover, to successfully hit a nail one has to use the hammer in the right way, i.e. not using the wrong side and not hitting the fingers accidently. Hence, knowledge of the right use and training are necessary.

The question about adequateness has to be asked for techno-pedagogical systems as well as for every teaching tool and method. What are the goals of a particular assessment? What is the benefit of tutoring? When is a brainstorming better than a moderated discussion? When and why did which method work and how can one implement it?

Those questions are systematically captured in design patterns which document tested solutions, reason about the problems solved and the contexts in which they can be applied: “Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution” (Alexander, 1979).

1.1.5 A pattern ontology

As we have seen, there are infinite patterns in the world but not every pattern counts as a pattern of good design and not every good design is captured in the literary form of design patterns. We can propose the following ontology of patterns:

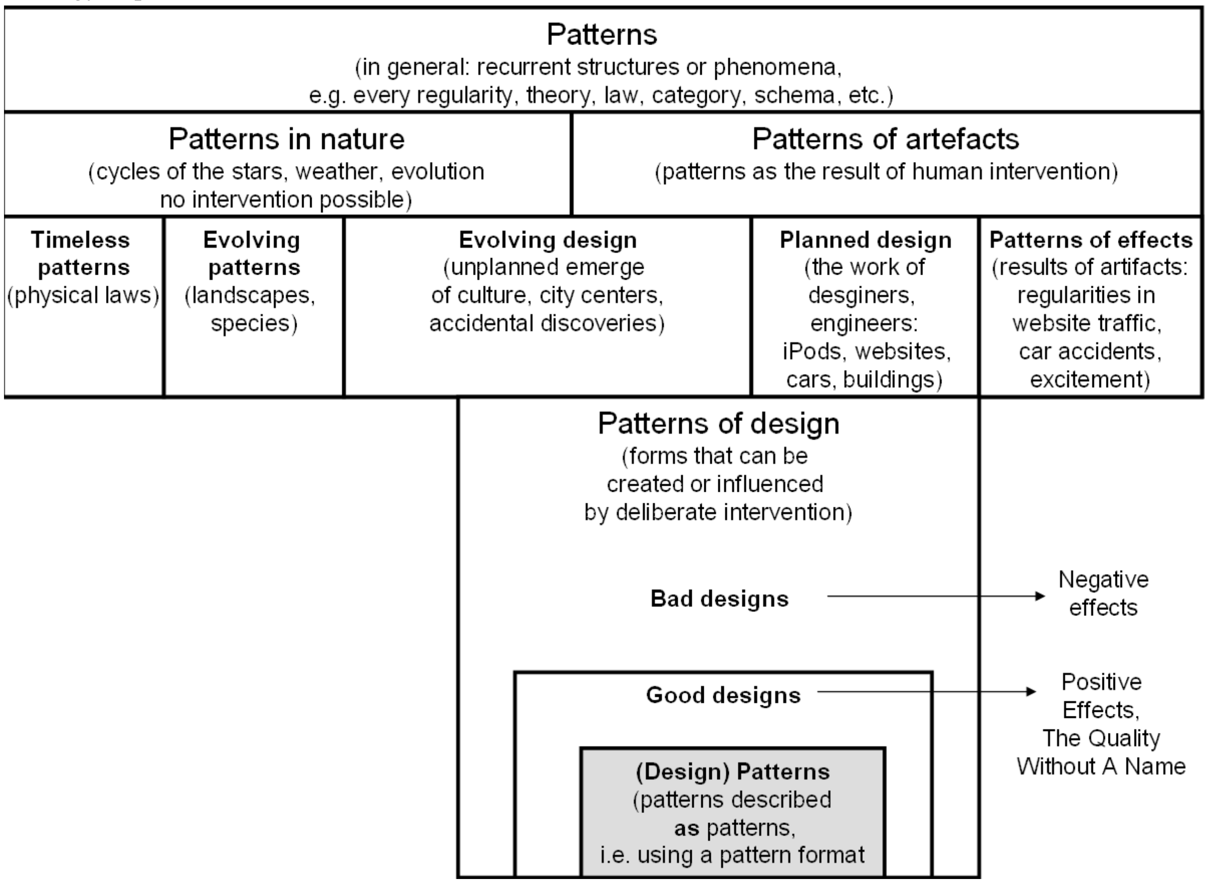

Patterns
(in general: recurrent structures or phenomena,
e.g. every regularity, theory, law, category, schema, etc.)
Patterns in nature
(cycles of the stars, weather, evolution
no intervention possible)
Patterns of artefacts
(patterns as the result of human intervention)
Timeless patterns
(physical laws)
Evolving patterns
(landscapes, species)
Evolving design
(unplanned emerge
of culture, city centers,
accidental discoveries)
Planned design
(the work of desginers, engineers:
iPods, websites, cars, buildings)
Patterns of effects
(results of artifacts:
regularities in website traffic,
car accidents, excitement)
Patterns of design
(forms that can be
created or influenced
by deliberate intervention)
Bad designs
Negative effects
Good designs
Positive Effects,
The Quality Without A Name
(Design) Patterns
(patterns described
as patterns,
i.e. using a pattern format

Figure 1. A pattern ontology

The focus of the theses will be on “good design” patterns and their documentation; however, many principles are valid for patterns in general and therefore we will consider general patterns as well. Only patterns in nature are not subject to our investigation. Natural patterns have been the study of science for a long time. Of course one can argue that the design of artefacts (according to human needs) is a natural process, too. H.A. Simon (1996) popularized the term “The Science of the Artificial” – and that is indeed what this work will be about: not theories about physical laws but theories about good design forms and good practices.

1.2 Benefits of the pattern approach

Capturing good patterns and describing them in written documents offers a lot of benefits. This section is a motivation why the pattern approach is important and invites the reader to consider whether the promises are interesting for her or his own field of work.

1.2.1 Solutions that improve our environment

The goal is to search for patterns that make a place or a design more whole, more alive. “The specific patterns out of which a building or a town is made may be alive or dead. To the extent they are alive, they let our inner forces loose, and set us free; but when they are dead they keep us locked in inner conflict” (Alexander, 1979, p. 101). Good solutions are liberating because they harmonize with human needs. A place or object that does not reflect your needs – that requires you to adapt – cannot be whole. A building that is in conflict with human needs cannot be a good design. Software code that is optimized for mathematical needs rather than human understanding can hardly be good because it is harder to maintain, more error-prone and uneasy to “live” in (Gabriel, 1996). An educational setting that tortures the students, the teacher or both cannot be a good solution.
Yet we all know situations and places where we feel free and comfortable – the coffee shop around the corner, a fireplace on a cold winter day, an office desk at the window. We have seen code that is beautiful. We remember

lessons in school that brought enlightenment, situations we got sucked into a moment of "Flow", or discussions that made us feel alive.

The configurations that raise such positive feelings in humans are good solutions, the patterns we are looking for. They have an organic order, "the kind of order that is achieved when there is a perfect balance between the needs of the parts and the needs of the whole" (Alexander, 1975). It is often forgotten that any design artefact has purpose and is not a means to its own end.

1.2.2 Pointing out serious problems

Very often we are not even aware of the problems that cause our sometimes miserable feelings about the world. A lecturer who tortures his students with overcrowded PowerPoint slides is certainly not aware that the amount of information should reflect the human capabilities of information processing. A conference without well planned coffee breaks ignores that attendees not only need a rest between the presentations and workshops but also need time to reflect and chat about the sessions in a comfortable environment. A programmer who designs a system that is not flexible for future adoptions will have a problem when the requirements change.

Without the awareness of such serious problems it is impossible to do something about them. To understand the problem in depth is the key to find an appropriate solution. "…patterns avoid rework that comes from inexpert design decisions. As an example, programmers who don't understand idioms like the Counted Body Idiom […] either will spend a long time converging on the solution, or will employ solutions that are less maintainable or just plain wrong" (Coplien, 1996). A pattern description highlights the problem and explains why the solution works. If we understand the problem, we can start to build the solution.

Experts very often do not only know good solutions but are more aware of common problems. Patterns are a way to explicate such tacit knowledge. "One important goal of the culture was make tacit knowledge explicit – especially that tacit knowledge kept in the heads of experts who had grass-roots experience" (Coplien, 2004).

1.2.3 Reusability and preservation of good solutions

If for a recurrent problem there is a good solution one does not need to reinvent the wheel. Design patterns are a toolbox from which a software designer or an educator can choose the right tools. To reuse successful designs does not mean that one just puts together ready-made components as one would do with Lego bricks. Patterns are not additive but multiplicative in the way they are combined. They are structural regularities that can overlap and that can be merged with each other. While design patterns are "are an aggressive disregard of originality" (Foote, 1997) in that they only capture existing solutions they are by no means a call against innovation or improving existing designs. Rather their objective is to not restart inventing the same old things again and again. The time and energy saved by using proven solutions can be used to develop new and innovative scenarios and adapt to the specific needs of a situation. Good forms are rare in comparison to forms that do not work. Therefore, we need to preserve those forms. Only if we have stable forms we can rely on, there will be opportunities for experimentation, trying out new things and evolving our designs. The combination of reliable forms to larger whole opens the door to infinite new ideas my means of composition.

1.2.4 Reducing complexity

Patterns help to reduce the complexity of design tasks in two ways. First, a complex system is split into smaller parts that can be addressed independently to some extent. Those smaller systems are loosely coupled to each other but they are not isolated because the context is always part of the analysis. By splitting the whole into parts (rather than composing the whole from isolated parts) one gets manageable parts that can be shaped independently. To develop a curriculum is reduced in its complexity due to the fact that while its components (e.g. lectures, seminars, projects, assessments) are interrelated each of them can be designed on its own.

The other reason why patterns help to reduce the complexity is that redundancies are made explicit. If we consider the educational environment as one large structure we will see that certain structural units recur – types of assessment, media formats, forms of collaboration etc. Those recurrent structures – the patterns – are redundant and to identify them helps to understand the whole system. For example, if it would be necessary to describe the complete structure of a lecture each time we describe a curriculum, the description of each learning or teaching setting would explode soon. We use the term "lecture" instead and assume that the structure of a lecture is known. Even if the term "lecture" is defined differently in diverse organisational contexts, we only need to describe once what is meant by "lecture" and then can refer to that meaning by using the term. To recognize and name recurrent structures is what we always do as the examples of the terms "lecture", "seminar", “assessment”, “homework” etc. show. All of these terms mean complex concepts. The description of design patterns very often cover structures where the terms and meanings are not yet part of our common language but experts and experienced practitioners are aware of them.

1.2.5 A shared vocabulary

In the domain of e-learning as for any other professional field there are certain forms and practices that are well known by experienced individuals who intuitively apply them. By means of an explicit description and denotation of such structures we can share knowledge between experts. At the same times the forms are ordered and classified (Baumgartner, 2006) and build a taxonomy (Baumgartner, 2011). Pattern descriptions help to harmonize mental models and to support the interdisciplinary communication between the agents who are involved in the design of educational settings. New forms, such as “e-assessment”[1] have already found a common term but the meaning is often interpreted differently between agents. People may have different pictures of what an e-assessment is. Just as definitions can be more or less adequate, pattern descriptions can be closer or more distant to a common understanding. What matters is that the explicit description of the denoted form can help to find a consensus about the meaning of a term as it is used in an interdisciplinary team. Erickson (2000) sees patterns as a means to build a lingua franca between interdisciplinary design teams.

1.2.6 Patterns that generate good design

Patterns are not fixed designs nor are they very abstract designs. Patterns are at a medium level of abstraction. Their form is not arbitrary but limited by structural constraints. Hence, a pattern must not be too general or too abstract in order to not loose the essential form. At the other hand, patterns are not too specific and avoid the dictation of exact steps. Hence, they are flexible and describe design spaces rather than single designs. For example, “vehicle” is not constitutive because it can mean very different things such as bike or an airplane. If somebody is asked to build a “vehicle” the output is not predictable. However, if we speak of a “car” it is clearer what is meant. Yet there are millions of ways to actually build a car. The pattern car is generative, real forms can be derived from that concept. The more specific patterns “cabriolet” and “van” still are generative for there are many variations of cabriolets and vans. A specific model of a car, however, is no longer a design pattern. Being fully specified there is no more variability. It is only a template and each instance created by that template looks similar except for surface properties such as colour. In the field of education the requirement to be generative could be interpreted that the term “test” is too abstract to specify a design. For example, if an e-learning designer is asked to implement a “test” this could mean almost anything. A “multiple choice” pattern would be more expressive. There are still a million creative ways to implement a multiple choice test but it is understood what kind of test is supposed to be implemented. The design of a test for driving licenses on the other hand has no space of variation at all because the questionnaires are standardized and each exemplar is only an instance based on a template.

1.3 Patterns in various domains

The pattern approach has its roots in the theory of architecture. Patterns should help individuals to identify and express their needs and requirements for the environments they live in (Alexander, 1979). In *A Pattern Language* there are 253 patterns for towns, buildings and constructions (Alexander 1979). Alexander first mentioned patterns in *Notes on the synthesis of forms* (Alexander, 1964) where he already introduced the concepts of fitness between context and solution, decomposition into sub-patterns and the reasoning for a form in terms of forces which is linked to the natural development of forms similar to the ideas described by D’Arcy Thompson (1942). Beck & Cunningham (1987) introduced the pattern concept to the field of user interface design and object-oriented programming. Patterns started to spread wide in the software community in 1994/95 when Gamma, Helm, Johnson & Vlissides (1995) published their standard work “Design Patterns: Elements of Reusable Object-Oriented Software” and Cunningham programmed the first Wiki ever to collaboratively write design patterns and share best practices.

Other disciplines started using the pattern approach, particularly

- **human-computer-interaction patterns:** interaction design (Borchers, 2001), user interfaces design (Tidwell, 2005; Raveh, 2005), computer mediated interaction (Schümmer & Lukosch, 2007), interface ontologies (Barr, Biddle & Noble, 2004), windows in user interfaces (Weir & Noble, 2004), design of widgets (Ingstrup, 2004), voice user interface design (Schnelle & Lyardet, 2007; Schnelle, Lyardet &

[1] The pattern has been described in the section „teaching scenarios“ on e-teaching.org

Wei, 2006; Schnelle-Walka, 2010), information retrieval (Schmettow, 2007; Lyardet, Rossi & Schwabe, 2000), interactive tabletops (Remy, Weiss, Ziefle & Borchers, 2010), user interface software (Coldewey, 1998), interaction design (Tidwell, 1998), interactive application (Wake, 1998), multimedia artefacts (Cybulski & Linden, 1998), time-based hypermedia artifacts (Lopes & Carriço, 2007)

- **web design and social community patterns:** websites (Van Duynie, Landay, & Hong, 2004; Mahemoff, 2006; Scott & Neil, 2008; Lyardet & Rossi, 1998), hypermedia systems (Garrido, Rossi & Schwabe, 1997), social interfaces, (Malone & Crumlish, 2009), wiki patterns (Mader, 2008), online communities (Homsky & Raveh, 2006), developer communities (Gabriel & Goldman, 1999), communities in collaborative systems (Schümmer, 2005b), knowledge sharing in online communities (Whitworth & Biddle, 2006)
- **organizational patterns:** organizational structures (Harrison & Coplien, 2005), change management (Rising, & Manns, 2005), group leadership (Homsky, 2004; Homsky, 2005), distributed product development teams (Bricout, Heliot, Cretoiu, Yang, Simien, & Hvatum, 2005), effective meetings (Schümmer & Tandler, 2008; Haase, 2006)
- **business patterns:** business strategies (Kelly, 2005; Kelly, 2006; Kelly, 2007; Kelly, 2008); customer relations (Weir, Noble, Martin & Biddle, 2005; Rising, 1997), process modelling (Kavanagh, 2005), innovation processes (Bienhaus, 2009), commercial use of open source (Link, 2009)
- **pedagogy:** incremental role plays (Eckstein, 1999), presentations (Reißing, 1999), learning to teach and learning to learn (Eckstein, 2001), experimental learning (Eckstein, Marquardt, Manns & Wallingord), active learning (Eckstein, 2003), course design (Bergin, 2001, Bergin, 2004), system-centred approach to learning (Bienhaus, 2004), teaching from different perspectives (Eckstein, Manns, Sharp, & Sipos, 2004), self tests (Steindl, 2001), seminars (Völter, Fricke, 2001), teaching software in classrooms (Schmolitzky, 2008), collective social learning (Brown, 2008), supervising thesis projects (Schmolitzky & Schümmer, 2008); mathematical learning (Mor & Winters, 2008); language learning (Köppe & Nijsten, 2012; Velázquez, 2012)
- **e-learning:** learning and teachings systems (Harrer, 2002; Harrer & Martens, 2005); designing learning management systems (Avgeriou, Retalis, & Papasalouros, 2004;Vogiatzis, Tzanavari, Retalis, Avgeriou & Papasalouros, 2005; Georgiadi, Retalis, Georgiakakis, 2009), e-learning in secondary education (Hadjisimou, Tzanavari, 2007), educational multimedia applications (Lyardet, Rossi, & Schwabe, 1998), interaction design for cross-cultural computer supported design learning (Schadewitz, 2008), intelligent tutoring systems (Salah & Zeid, 2009), online trainings (Kohls, 2009), screencasting (Chen & Rabb, 2009)

The original pattern idea evolved in the various domains and different interpretations emerged. Some communities kept closer to the texts of Alexander, others became more emancipated. A pattern culture developed (Coplien, 2004) and the primary events for presenting, sharing and improving patterns are now the Pattern Languages of Programs (PLoP) conference series.

1.4 E-Learning and educational patterns

All in all, the milieu in which the pattern approach is most successful is that of programmers and engineers. In this environment, the first pedagogical patterns have been developed as well, for example the "Patterns for Classroom Education" (Anthony, 1996). The Pedagogical Pattern Project was founded in 1995 at an OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) conference. Many pedagogical patterns have been collected on the project home page (http://www.pedagogicalpatterns.org/) and discussed in Writer's Workshops. Pedagogical patterns first boomed at the European Pattern Languages of Program conference (EuroPLoP) in 2000 when there have been several submissions on that topic for the first time. However, a Pattern Languages of Teaching conference, a EuroPLoT as proposed by Quiberly-Cirkel (1999b), was never held. Rather, the EuroPLoP hosted special Writer's Workshops on pedagogical patterns in the last years. Also, various scientific workshops about e-learning patterns have been co-located on international conferences such as CSCL 2003, EDMEDIA 2004, INTERACT 2005, CSCL 2007, GMW 2007, DeLFI 2008. Several research projects (including idSpace, TELL, E-LEN project, Pointer, PADI project, Pattern Language Network) have collected patterns and evaluated various methodologies. Goodyear, de Laat & Lally (2006) developed a pattern language for networked learning. Besides the scientific interest, several repositories with pedagogical patterns have been created in the last years, for example a collection of patterns for mathematical games (Mor & Winters, 2008), patterns that document didactical knowledge for higher education (Vogel & Wippermann, 2005), the E-Learning Design Patterns Repository and the Pattern Language Network (Finlay, Gray, Falcone, Hensman, Mor & Warburton, 2009). The popular German portal e-teaching.org which addresses teachers in higher education has started to collect e-learning patterns as well (Kohls, 2009b). Furthermore, there has been a special issue of Computers in Human Behaviors (Dimitriadis, Goodyear, & Retalis, 2009). Last but not least several PhD theses have been written about e-learning patterns (Köhne, 2005; Derntl, 2006, Wippermann, 2008, Zimmermann, 2008).

Recurrent interrelated structures – that is patterns - can be found on all levels and in different domains: at the level of organizational forms, the design of curricula, learning and teaching scenarios, course types, methods, tools, design of lessons and resources, the implementation of collaborative settings, interaction scripts, or the architecture of learning environments.

1.5 Challenges and research questions

1.5.1 Multiple meanings of "pattern"

We have already seen in the ontology of patterns that there are many different types of patterns. But things are even more complicated. Even in the context of design patterns, there are many different ideas about what patterns are and what the term "pattern" means (Vlissides, 1998). The term "design patterns" often means patterns of a certain granularity in the scope of software design (Buschmann, Henney & Schmidt, 2007). However, pedagogical patterns and HCI patterns are about design as well and frequently referred to as design patterns. The concept is very broad and almost anything could potentially be a pattern (Harrison, 1998). Johnson (1998b) points out that the *World Book Dictionary* provides 11 definition of the general term. We can find as many definitions for the more specific term of design patterns.

Sometimes, pattern_1 refers to "a generic solution to some system of forces" (Alexander, 1979, p. 147); another time pattern_2 means the pattern description (a "literary form of software engineering problem-solving discipline", Appleton, 2000; Gabriel (2002) refers to patterns as a text genre). Pattern_3 could also mean a specific implementation of a pattern. Moreover, Alexander distinguishes between general patterns_4 (as in *A Pattern Language,* Alexander, 1979) and particular patterns_5 of structure without recurrence (Alexander, 1964). At other times patterns_6 are considered as "rules" to create that structure or as relations between form and context ("…every pattern we define must be formulated in the form of a rule...", Alexander 1979, p.253). Hence, patterns are seen as both a thing and a generative process (Appleton, 2000).

If that is not confusing enough, there are also different schools or flavours of the pattern concept (Buschmann, Henney, & Schmidt, 2007). A pattern means different things (structures in the world, invariants of those structures, the description of the invariants etc.) and there are different interpretations what these things are. For example, the various fields have established their own description formats, styles of writing and modes of reasoning. This is a challenge – or the benefit of diversity – that has been long known in the pattern community: "Patterns are expressed in many forms and styles. A search for a clear and complete definition of a pattern may very well be an elusive endeavor. A room full of pattern experts may come to a consensus on which patterns are good, which patterns contain the 'quality without a name', or if they even are patterns. It is harder for these experts to consistently define what a pattern is and what makes it good" (Corfman, 1998). There is not a single theory about patterns, rather a pattern theory_1, pattern theory_2, …, pattern theory_n.

The research question RQ-1 is: What is the nature of patterns and pattern languages?

1.5.2 Meaningful patterns

The world is full of patterns but not every pattern is worth to be documented. For example, for a pattern_1 that is well known in a culture (such as a TABLE) it seems to be inappropriate to write it down as a pattern. Yet there are many badly designed tables in restaurants, on conferences or in furniture stores that we could see a reason to document the needs (forces) of a harmonic table. But what is the meaning of a TABLE? There are so many different tables for different contexts (tables for living rooms, night tables, tables in restaurants, etc.) that it is hard to find a meaningful core. "To put things on it" does not work as a precise description because then we could consider a chair *as* a table but that is not the same as *to be* a table. It is not the nature of a chair to be a table and chairs are hardly tables that are comfortable. TABLE is too abstract to be a meaningful pattern – more precisely to provide a meaningful *pattern description*. Meaning should also be the guiding force in the development of software (Qullien, Rostal & West, 2009). Software and patterns need to serve a purpose (Olson, 1998).

Another question arises about "trivial" patterns. A table is not a trivial but a well known solution (it is trivial to select a table if you want to put something on it but the form is not trivial). But there are also trivial solutions to serious problems that are not well known. For example, it is trivial to put a "Do not disturb" sign at your door during an online conference. Yet most people do not use it because they are not aware that it could be a problem if another person just stumbles into the room and disturbs them. The solution is simple but meaningful. A good measure for a pattern should be the problem that is solved (Richards, 1998).

Another level of meaning involves different target groups. An abstract pattern can be meaningful to someone who has experience in particularizing the abstract – if we know what a brainstorming is, we can describe a complex teaching scenario by saying: "The lesson starts with a brainstorming, followed by…" Yet if a novice does not know how to run a proper brainstorming this description is too abstract; it cannot be made alive because the reader misses what a brainstorming is. The meaning of a pattern, hence, depends on the experience of the target audience and whether it solves problems that are important to that audience.

The research question RQ-2 is: What makes patterns meaningful?

1.5.3 The right level of abstraction

In education we find very often methods and models that are either too abstract or too specific (Baumgartner, 2011). A pattern that is too abstract might become meaningless because the relevant parts are missing. What is relevant depends on the context and the experience of the person who applies the pattern. A very detailed description on the other hand is more specific but it is less open for variations in actual contexts. Patterns should provide enough detail but be general enough at the same time (Coplien, 1996). It is often said that patterns are on a medium level of abstraction (Gabriel, 1996). But "medium" is a wide range. Rising (2007) points out there might not be one right level of abstraction. The challenge is that patterns can be too vague or contain too many details (Buschmann, Henney, & Schmidt, 2007).

The research question RQ-3 is: What is the right level of abstraction?

1.5.4 Granularity of patterns

Pattern descriptions of a lower level of abstraction lead to an explosion of documents if each variation is covered in a dedicated pattern document. One way out is to combine the extracted patterns with each other. A pattern that tackles organizational aspects of an online meeting could be outsourced in order to avoid redundant descriptions for the online training and the online presentation. The outsourcing of parts of a pattern or the splitting of a pattern into sub-patterns has the advantage that the description of a pattern does not become too bulky. Moreover, it is easier for such outsourced patterns to identify further contexts of application. "Prepared examples" is a common pattern not only for online trainings (see appendix F for the pattern language) but for other trainings as well. One could describe how to prepare a good example as part of the pattern description "online trainings". But then we could not reuse it for other training forms.

In principle each part that recurs could be outsourced into a single pattern. However dividing patterns into too small units may result in patterns that are trivial. For example, the pattern "Do not disturb" reasons that when an online training runs one should put a sign at the door signalling that there is an event that should not be disturbed. It is very small and one can ask whether it is worth the effort to write a single pattern for it.

The organization of patterns into collections or languages has been a topic in the pattern community for a long time (Harrison, 1998). Alexander (1964) provides suggestions on parting a larger whole into smaller wholes based on the connections between elements. It is a frequent suggestion in Writers' Workshops that one pattern could be split into several smaller ones. This seems to be a general experience in the pattern community (Rising, 1998). Buschmann, Henney and Schmidt (2007) name several examples where one pattern consists of several smaller patterns or one pattern description tackles different patterns.

The research question RQ-4 is: What is the right size of granularity?

1.5.5 Objectivity

To have patterns on different levels of abstractions and of different size and scope leads directly to the question how objective a perceived pattern is. If some solutions do not vary very much it is easier to denote a category everyone agrees on. If there are only some commonalities it gets harder to agree, e.g. a lecture can have many forms and people may not only have different ideas about which form a lecture should take but also deny for an extravagant lecture that it is a lecture at all. If somebody says "That was not a lecture!", s/he may not only express a distaste for the quality of the lecture or that it was a bad exemplar of the category but rather that it was an exemplar of another category altogether, i.e. in the sense of "that was more like a quiz show than a lecture."

The function of an object is supervenient to its form, i.e. the functional properties depend on the form properties. Functional requirements are less objective because different individuals may care for different functions and values. What functions people expect from a car (family trips, buy food, or speed rallies), a cell phone (phoning, listen to music, make photographs), or a software application can vary. The "Quality Without a Name" and its wholeness are aesthetical properties (Coplien, 1996). Are these values objective, normative, or individually constructed? For educational patterns it might be harder to find patterns everyone agrees on because there are different schools of thoughts and different beliefs about what is good teaching. Furthermore, not all documented patterns are good (Coplien, 1998d) and it is a general challenge to know whether a pattern is a good one (Harrison, 1998).

The research question RQ-5 is: Can we objectively describe patterns?

1.5.6 Pattern mining

There are many ways to find interesting new patterns. We can draw from our own experience, ask colleagues and experts, or investigate existing systems (DeLano, 1998). However, pattern descriptions go deeper than describing recurrent solutions (Beck, Coplien, Crocker, Dominick, Meszaros, Paulisch & Vlissides, 1998). They provide reasons for the solution (forces) and name the contexts of applications. These are much harder to find because they cannot be seen directly.

There is also a good chance that the patterns one person sees are not the patterns another person sees. People have different experiences and may focus on different things. There is a need to find patterns that are shared and accepted by other practitioners.

The good solutions, or nuggets of wisdom, need to be documented in an accessible way. To share knowledge means to communicate information. If the vehicle of communication is distracted by noise (e.g., unclear language) it is much harder or impossible to get the right picture of the pattern.

The research question RQ-6 is: How do we mine patterns effectively?

1.5.7 Old wine in new bottles

There are at least two reasons to see patterns as old wine in new bottles. First, the sharing of best practices is not exclusive to the pattern approach (Booch, 1998). In pedagogy the description of pedagogical models, methods and practice is not new (Wedekind, 2011). So, are patterns just a way to package these old models and methods in a new suit? Where is the difference of the pattern approach to collections of methods, design principles and guidelines?

The other reason is that patterns are supposed to capture existing proven solutions. As a result, they only describe things that already exist (Coplien, 1996; Harrison, 1998). They do not present any novel ideas or propose solutions to unsolved problems. Is there any scientific progress in writing down a pattern? Is it appropriate to publish patterns in scientific journals or can we only publish reports about pattern application?

The research question RQ-7 is: What is novel and original in the pattern approach?

1.5.8 Misapplication of patterns

People have often abused the concept of patterns in many ways (Coplien, 2004). The first misunderstanding is that people think that patterns automatically improve designs, i.e. the more patterns are used, the better the design (Corfman, 1998). However, it is not the quantity of patterns but the application of the right (appropriate) patterns that make a design good. A hammer is a good tool only for specific tasks. Likewise, each pattern is appropriate for certain contexts but not a silver bullet. Köppe (2011) has developed patterns about teaching patterns and he highlights in UNDERSTANDING DESIGN PATTERNS that it is important that students understand to choose the fitting pattern.

Another kind of misapplication is the wrong particularization (Gabriel, 1998). If we have identified the right pattern to solve our problem, we have to adopt it to the specific situation. For example, the way parameters are handled in an OBSERVER could look very different than in the examples of the documented pattern. The number and names of methods are always varying. If observing objects are notified via a PROXY, the class structure looks different to the diagrams given in the pattern description; yet the structure is still an OBSERVER.

Due to the success of patterns people have jumped on the bandwagon and started to use the pattern format to document ideas that are not patterns. This includes untested proposals as well as forms that are too abstract to have meaning, too specific to allow adoptions for new designs, or miss important parts of a whole.

The research question RQ-8 is: How to avoid misapplications of patterns?

1.6 Research methods and chapter overview

This thesis hopes to resolves the challenges in the next chapters. We will capture the state-of-the-art in chapter 2 which summarizes the current understandings about what patterns are, where the roots are and which values exist in a pattern culture. Furthermore, a framework will clarify the relations between patterns in the world, patterns in our heads and patterns in written form. The theoretical foundation for the framework will be developed in chapter 3 by linking the pattern approach to established theories from other fields. The framework will be supported empirically by the author's work on patterns in chapter 4. The implications for the pattern approach and for pedagogical research are highlighted in chapter 5.

1.6.1 Overview of chapters

Chapter 1: Introduction
This chapter provides an overview about the pattern approach, its benefits and challenges. Based on the challenges research questions are derived.

Chapter 2: State of the Art: The nature of patterns
Chapter 2 is an attempt to clarify which different understandings and interpretations of the pattern approach exist. We will investigate the meaning of context, problem, forces, solution, and consequences. The concepts will be illustrated with a path metaphor and followed by a discussion. To understand the quality of patterns we will reason about wholeness, proper encapsulation, mid-level abstraction, composability, openness, generativity, equilibration and fitness, evolution, meaning and ethics. The value system is based on the understanding of the (software) pattern community. Therefore, we will have a look at the emergence of its culture and the rituals installed to achieve high quality patterns: pattern conferences, shepherding of papers, and Writers' Workshops. We will also take a look at current discussions and how Alexander's later work "The Nature of Order" is perceived and how it relates to the pattern approach.

Chapter 3: Theoretical framework: Theories related to patterns
Chapter 3 develops a theoretical framework for understanding different views on patterns: patterns in the world, patterns in our heads and explicated pattern descriptions. It links the pattern approach to theories of other domains in order to get insights to the nature of patterns and resolve many of the challenges. To understand patterns in the world we will consider theories of forms, information and abstraction; to understand patterns in our heads we will explore the similarities to schema theory and tacit knowledge; to understand pattern descriptions and their validity we will argue that principles from the theory of science (epistemology) can be borrowed because the descriptions make explicit and testable statements about our world. The framework in this chapter is based on a literature review of existing theories of other domains, feedback from Writers' Workshops, a focus group on pattern mining and a thought experiment. The data is integrated into a coherent theoretical framework.

Chapter 4: Empirical Support: Practices of Patterns
Chapter 4 provides empirical support for the framework developed in chapter 3. Each subsection in chapter 3 has its counterpart in chapter 4. To show how patterns in the world are generated we will use the lessons learnt from a case study about a concept implementation of a software system and the pattern mining of interactive graphics. To test whether patterns are induced in our heads according to schema theory we will discuss the results from laboratory studies with several groups. To investigate the empirical content in pattern descriptions we will have a retrospective document analysis of several pattern descriptions. We can compare the information content by contrasting different versions of pattern descriptions and pattern formats.

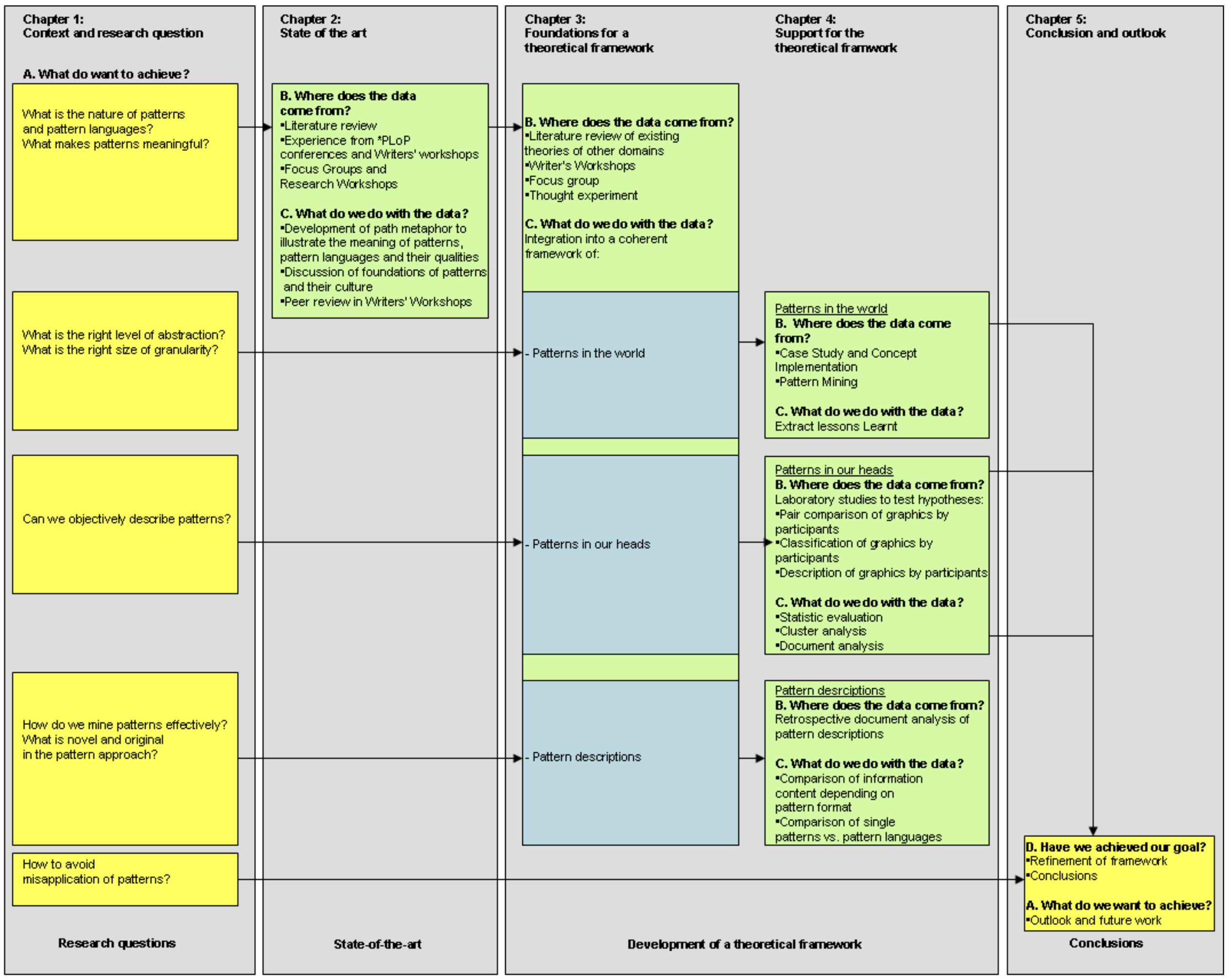
Chapter 5: Conclusions and Future Research
Chapter 5 returns to the challenges and tries to address them based on the exploration given in chapter 3 and 4. We will discuss the implications for the pattern approach and argue why the pattern approach is so promising for the field of education and e-learning. An outlook to further research in the future will finish this work

1.6.2 Overview of research methods used in the chapters

Holz, Applin, Haberman, Joyce, Purchase & Reed (2006) have developed a general framework for computer research methods which is grounded in four questions:

A. What do we want to achieve?
B. Where does the data come from?
C. What do we do with the data?
D. Have we achieved our goal?

The questions can be asked iteratively, i.e. asking about the goal achievements can trigger new research questions. Following this framework we have defined the context and research questions (A.) in chapter 1. For each of the research questions we can identify research methods to collect (B.) and analyze (C.) data in chapter 2-4 to address the research questions. A reflection of goal achievement (D.) and emerging new questions (A.) can be found in chapter 5.



Chapter 1: Context and research question
A. What do want to achieve?
What is the nature of patterns and pattern languages? What makes patterns meaningful?
What is the right level of abstraction? What is the right size of granularity?
Can we objectively describe patterns?
How do we mine patterns effectively? What is novel and original in the pattern approach?
How to avoid misapplication of patterns?
Research questions
Chapter 2: State of the art
B. Where does the data come from?
•Literature review
•Experience from *PLoP conferences and Writers' workshops
•Focus Groups and Research Workshops
C. What do we do with the data?
•Development of path metaphor to illustrate the meaning of patterns, pattern languages and their qualities
•Discussion of foundations of patterns and their culture
•Peer review in Writers' Workshops
State-of-the-art
Chapter 3: Foundations for a theoretical framework
B. Where does the data come from?
•Literature review of existing theories of other domains
•Writer's Workshops
•Focus group
•Thought experiment
C. What do we do with the data?
Integration into a coherent framework of:
- Patterns in the world
- Patterns in our heads
- Pattern descriptions
Chapter 4: Support for the theoretical framwork
Patterns in the world
B. Where does the data come from?
•Case Study and Concept Implementation
•Pattern Mining
C. What do we do with the data?
Extract lessons Learnt
Patterns in our heads
B. Where does the data come from?
Laboratory studies to test hypotheses:
•Pair comparison of graphics by participants
•Classification of graphics by participants
•Description of graphics by participants
C. What do we do with the data?
•Statistic evaluation
•Cluster analysis
•Document analysis
Pattern desrciptions
B. Where does the data come from?
Retrospective document analysis of pattern descriptions
C. What do we do with the data?
•Comparison of information content depending on pattern format
•Comparison of single patterns vs. pattern languages
Development of a theoretical framework
Chapter 5: Conclusion and outlook
D. Have we achieved our goal?
•Refinement of framework
•Conclusions
A. What do we want to achieve?
•Outlook and future work
Conclusions

Figure 2. Overview of methodology and how each chapter addresses the research question

RQ-1: What is the nature of patterns and pattern languages?
In chapter 2 data is collected by literature reviews, based on ethnological experience from the *PLoP conferences (Writers' Workshops, Shepherding Workshops, Sheep Workshops, Bootcamps) and participation in or organization of focus groups and research workshops about patterns. The data is used to develop a path metaphor that directly shows the structural properties of patterns. Writers' Workshops are seen as a scientific method (Gabriel, 2008a) to refine work on patterns. The path metaphor has been further developed based on the feedback from Writers' Workshops at PLoP 2010, PLoP 2011 and EuroPLoP 2012.

RQ-2: What makes patterns meaningful?
To follow this research question we will explore qualities that have been frequently named in the pattern community; based on a literature review we can illustrate some of these qualities for the path metaphor. We will also discuss the foundation of patterns by having a closer look at Christopher Alexander's original work and link

it to gestalt theory. To better understand the values of the pattern community we will trace back the history of the pattern community.

RQ-3: The research question is: What is the right level of abstraction?
To answer this question we will create a fictive world structure for a thought experiment in chapter 3.2. The structure can be used to show how different types of abstraction work on parts of that structure. We will explore the relation between form, information and abstraction. In the empirical counterpart (chapter 4.2) we will discuss the implementation of a software system that creates pattern instances of interactive graphics. The catalogue of available patterns in that system reflects the choices of abstracting over existing interactive graphics. We will discuss the lessons learnt from the generation process of that system and the challenges in the mining process.

RQ-4: What is the right size of granularity?
This research question is closely related to the previous one and will be tackled by the same approaches: a thought experiment in 3.2 and the lessons learnt from a case study about a software system in 4.2. The focus will be on considering how the amount of information in a pattern shifts depending on its granularity. This can be explored for the fictive world structure and shown for the pattern descriptions analyzed in 4.4.

RQ-5: Can we objectively describe patterns?
To reflect on this question we will consider how patterns are mentally constructed according to schema theory in chapter 3.3. The empirical part in 4.3 consists of several experiments to find out whether patterns are indeed mentally induced as schema theory suggest and whether individuals construct the same schemas. Each phase of the experiment is guided by more detailed research questions introduced in the respective sections.

RQ-6: How do we mine patterns effectively?
We will consider common practices of pattern mining in the light of epistemological goals. The implication of induction, deduction and abduction are discussed for the pattern approach. Chapter 3.4 will present typical mining methods that are used in the pattern community based on literature review and a focus group that was held at a EuroPLoP conference. A retrospective analysis of pattern descriptions in chapter 4.4 will show that patterns and the statements found in patterns are indeed subject to empirical falsification.

RQ-7: The research question is: What is novel and original in the pattern approach?
The section on the epistemology (3.4) of patterns will reason about the amount of knowledge (content) provided in a pattern description. The scope of content depends on the size and universality of a pattern as well as the questions one asks about a design. Chapter 2 shows that the pattern approach offers a coherent framework of asking questions about the context, problem, solution and consequences of a design. Chapter 4.4 analyzes pattern descriptions to compare the information content of different description formats. It also compares the information that is found and required in single pattern vs. pattern languages.

RQ-8: How to avoid misapplications of patterns?
The framework developed in chapter 3 and 4 should clarify what the nature of patterns is and provide answers for practical guidance for identifying the right patterns, documenting patterns in the right way, selecting patterns in the right way, and implementing patterns in the right way. Misapplication can be avoided if we have a good understanding of what patterns are, what benefits they offer and which limitations they have.

1.6.3 PLoP and EuroPLoP submissions and the discussion in Writers' Workshops

The participation in Writers' Workshops has been an important activity during the writing of this thesis. Parts of this work have been published during this process and have benefited from the feedback of peer reviewers, shepherds and Writers' Workshop participants. The idea for the path metaphor (chapter 2) was born early in 2009 when I was working on a special content area for e-teaching.org about e-learning patterns. The metaphor has been continuously refined and clarified based on feedback from the pattern community. The metaphor has been submitted to PLoP 2010 (Kohls, 2010b) with a focus on the general structure and to PLoP 2011 (Kohls, 2011a) with a focus on the qualities of patterns. A version that integrated the comments and focussed on both was submitted to EuroPLoP 2012 (Kohls, 2012c). Chapter 2 also includes materials from an international e-learning pattern workshop at the Knowledge Media Research Center (Tübingen, Germany) organized by Kohls & Wedekind (2009). The revised submissions have been published in an edited book (Kohls & Wedekind, 2011a). Likewise, my views on the psychological and epistemological status of patterns have been discussed in the pattern community. Chapter 3.3 about patterns in our heads is based on a PLoP 2008 submission (Kohls & Scheiter, 2008). Chapter 3.4 about the description of patterns is based on a PLoP 2009 submission (Kohls & Panke, 2009). All of the author's pattern descriptions that are discussed in this thesis have been submitted to PLoP or EuroPLoP conferences (see appendix E and F for the latest versions). The empirical support in chapter 4 about generating patterns with an authoring tool, testing patterns with an experimental setup and evaluating the content of patterns by comparing different versions have not been published before. While some data of phase 3 of the experimental setup has been published (Kohls & Uttrecht, 2009), the discussion in this thesis tackles more research questions, covers all phases of the experimental setup and is based on further data.

2. The nature of patterns and pattern languages

2.1 Motivation

This chapter aims at learning more about the structure of patterns by examining a very simple and intuitive example of a pattern: a path in a hillside landscape. I have found it helpful to use the metaphor of proven paths in a territory to explain how patterns capture, describe and explain good practices. A path is an example of a solution that leads to a specified goal and it is a simple concept that is very well known to everyone. This simplicity enables us to show how the territory and the overall situation define which paths are potential solutions and which properties each path has.

Pattern sceptics very often do not understand what is innovative about the pattern format. Best practices have been documented for a long time and scientific discovery is all about finding invariants – patterns – in nature. While that is true, the elaboration in the pattern format does not only describe recurrent forms but reasons about the form in terms of forces and consequences. Most people agree that a solution has to be adequate for the task at hand. However, many descriptions that do not follow one of the pattern formats lack this context information. The purpose of this chapter is to use a metaphor to address common sense. Nobody would follow directions that belong to a different landscape. Nobody is happy about directions that are too abstract (e.g. missing one or two turns) or too detailed (e.g. describing every stone of a path). Nobody would follow directions unless they make sense, i.e. it should be reasonable to follow a path. Yet many descriptions of solutions do not provide this information. They are too abstract or detailed and fail to explain why a form solves a given problem. Patterns, however, capture contexts, forces, consequences and solutions at a medium level of abstraction.

Pattern beginners who are willing to share their knowledge by writing patterns will see that it takes more than just a knowing solution to describe it properly. Once we are aware of something that has worked several times the scholastic mind is eager to write it down. But it is much harder to write down why something has worked and to reason under which circumstances the configuration will work or fail. The path metaphor hopefully shows that it is worth the effort. By explaining a solution form along with the fitting environment and what has shaped the solution we can help the user to understand the form and make sense of it. In order to adapt a pattern to a specific situation one should know what has influenced the general solution.

Pattern experts can use the visualizations to discuss their conceptions about patterns. Both agreement and disagreement will give new insights to how we see patterns. Since human beings are very good at perceiving visual patterns, some properties of recurrent structures might be grasped more intuitively, e.g. how patterns can be divided into sub pattern, how patterns overlap and how patterns work together.

2.2 Definitions

2.2.1 A basic definition

Let us begin with Christopher Alexander's most cited definition of patterns in *A Pattern Language*:

„Each pattern describes a **problem** *which occurs over and over* again in our **environment**, and then describes the core of the **solution** to that problem, in such a way that you can use this solution a million times over, *without ever doing it the same way twice*." (Alexander, Ishikawa, Silverstein, Jacobsen, Fiksdahl-King, Angel, 1977)

There are two parts of this definition that are essential to patterns:

1. a pattern is the solution to a problem in an environment
2. the form of the solution can be used over and over again without ever doing it the same way twice

Both statements, simple as they may appear, are rich in their meaning. Let us therefore consider a familiar environment, a hillside landscape, to explore the concept of patterns. In such an environment, if we intend to hike from one place to another, we may encounter the problem that we cannot directly move in a straight line to the destination because we cannot cross canyons or pass over steep walls. Each trail that guides us to the destination is therefore a solution to our problem: getting us from where we are to our desired destination.

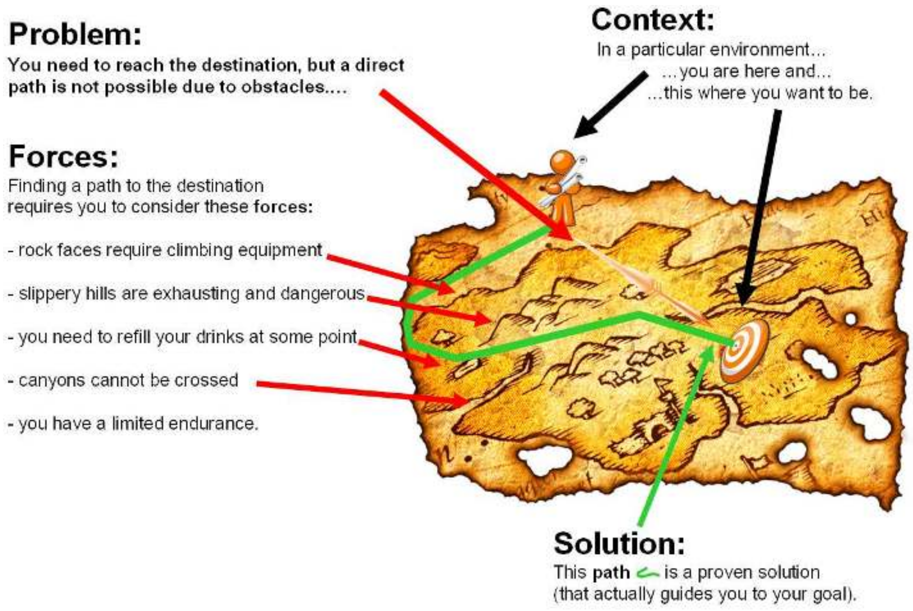

Problem:
You need to reach the destination, but a direct path is not possible due to obstacles....
Context:
In a particular environment...
...you are here and...
...this where you want to be.
Forces:
Finding a path to the destination requires you to consider these forces:
- rock faces require climbing equipment
- slippery hills are exhausting and dangerous
- you need to refill your drinks at some point
- canyons cannot be crossed
- you have a limited endurance.
Solution:
This path is a proven solution
(that actually guides you to your goal).

Figure 3. Context, problem, forces and solution of a path.

2.2.2 Definitions by Alexander

The deepest elaboration of Alexander's understanding of patterns can be found in *The Timeless Way of Building* (Alexander, 1979). We will explore some statements that define aspects of patterns and discuss their implications.

Alexander points out patterns of relationships constitute larger structures: "A large part of the 'structure' of a building or a town consists of patterns of relationships" (Alexander, 1979, p. 87). Hence, the term pattern always refers to relationships. Those relationships are found in space and define a recurrent form: "Each of these patterns is a morphological law, which establishes a set of relationships in space" (Alexander, 1979, p. 90). Each set of relationships – each structure – has sub sets of relations – sub structures: "And each law or pattern is itself a pattern of relationships among still other laws, which are themselves just patterns of relationships again" (Alexander, 1979, p. 90). Patterns are always recursive: patterns are made of smaller patterns in the same way as large structures consist of sub-structures: "The individual configuration of any one pattern requires other patterns to keep itself alive" (Alexander, 1979, p. 131). We will discuss the related meanings of patterns, structures, relations, and forms in section 3.1.

While any recurrent structure is a pattern, we are only interested in valuable patterns: "Good patterns are good because to some extent each one of them reaches the quality without a name itself" (Alexander, 1979, p.116).

In his later volumes, *The Nature of Order*, Alexander focuses on the "degree of life" that a structure has. In *The Timeless Way of Building* he acknowledges this quality feature already: "The specific patterns out of which a building or a town is made may be alive or dead. To the extent they are alive, they let out inner forces loose, and set us free; but when they are dead they keep us locked in inner conflict" (Alexander, 1979, p. X).

If patterns can be dead or alive then we are looking for the positive ones. Such patterns are both generic and generative (Buschmann, Henney & Schmidt, 2007) as the following two statements suggest:

> "Each pattern is a generic solution to some system of forces in the world" (Alexander, 1979, p. 147).
>
> "Each pattern is a rule which describes what you have to do to generate the entity which it defines" (Alexander, 1979, p. 182).

This has led the pattern community to see patterns as both a thing and process (Appleton, 2000). While patterns are consequently referred to as laws or invariants, Alexander also calls them "rules of thumbs" (Alexander, 1979, p.202). The written patterns are explications of the structure: “To make a pattern explicit, we merely have to make the inner structure of the pattern clear” (Alexander, 1979, p. 249). This explication is as hard as any science, and each statement about the pattern is vulnerable to empirical data. The reasons for the structure of patterns are due to sets of interrelated forces. A diagram of these forces implies the structure of the pattern: “These diagrams [of forces], which, in my more recent work, I have been calling patterns, are the key to the process of creating form” (Alexander, 1964, preface to 2nd edition).

The explicated patterns are described in a specific format that includes a discussion of the forces, the context in which they arise and the structural configuration to resolve them: “…every pattern we define must be formulated in the form of a rule which establishes a relationship between a context, a system of forces which arises in that context, and a configuration which allows these forces to resolve themselves in that context” (Alexander, 1979, p.253).

The nesting of patterns and their interrelations is focussed in Alexander’s later work where he speaks of centres instead of patterns:

„From the point of view of relationships which appear in the design, it is more *useful* to call the kitchen sink a ‘center’ than a ‘whole’. If I call it a whole, it then exists in my mind as an isolated object. But if I call it a center, it already tells me something extra; it creates a sense, in my mind, of the way the sink is going to work in the kitchen. It makes me aware of the larger patterns of things, and the way this particular element – the kitchen sink – fits into that pattern, plays its role in that pattern” (Alexander, 2002a, p.85).

2.2.3 Definitions of the software community

The software community draws widely from these ideas. However, very often they are trivialized. On the official website of the software pattern community, we can find the comment from Richard Gabriel[2]:

> “Various people write things like this:
>
> ‘A pattern is a proven solution to a problem in a context.’
>
> I suppose I cannot argue with the actual words, because they are not obviously false, but I fear that among the software crowd, especially the CS-oriented ones, this represents a misconception.
>
> Alexander writes:
>
> ‘Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.’
>
> I'm sure that because he wrote this many feel that the rewording above is a fair copy. I don't think it is. Alexander is capable of writing simple sentences when the thought he wishes to express is simple. He could easily have said "a pattern is solution to a problem in a context" - he uses most of these words already.
>
> But, instead he wrote the paragraph above and he went on to explain:
>
> ‘As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves.
>
> As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant.’”

Therefore, Gabriel suggests the following definition for the field of software design:
“Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves.”

[2] http://hillside.net

The goal of writing and using patterns is also manifest on the start page of Hillside.net:
"Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience."

As we have seen in Alexander's definition, a pattern is both a thing and a process. For software design this means that "the term pattern applies both to the thing (for example, a collection class and its associated iterator) and the directions for making a thing. In this sense, software patterns can be likened to a dress pattern: the general shape is in the pattern itself, though each pattern must be tailored to its context." (Coplien, 1998d)

Buschmann, Henney, & Schmidt (2007, p. 348) summarize in *POSA-5* the common core properties of patterns and pattern languages:

> "Each defines a process and a thing."
> "Each ensures that the 'thing' whose creation it supports is of proven and sound quality in its structure, behaviour, and core properties."
> "Each explicitly addresses the forces that can influence the concrete look-and-feel of the designs and the character of the environments that it helps to build."
> "Each supports genericity by describing entire solution spaces, not (just) single point solutions."
> "Each enables communication, discussion, and dialog about its subject to aid understanding of the nature, forces, and consequences of the solution presented."
> "Each has a concrete name so that its audience can remember it and immediately know what subject or domain it addressed."
> "Each is a work in progress that should constantly be updated toward new and evolved experiences and knowledge in its subject domain."
> "Each can be implemented in many different ways without ever being 'twice the same' – dependent on the concrete requirements, constraints, and scope of the systems or environments in which it is applied."

2.2.4 Definitions of the pedagogical and e-learning community

The field of education comes with its own interpretation of Alexander's ideas (Derntl, 2006) but is largely influenced by the software community (Köhne, 2005). On the Pedagogical Patterns Project homepage, we find the following definition:

"Patterns are designed to capture best practice in a specific domain. Pedagogical patterns try to capture expert knowledge of the practice of teaching and learning. The intent is to capture the essence of the practice in a compact form that can be easily communicated to those who need the knowledge. Presenting this information in a coherent and accessible form can mean the difference between every new instructor needing to relearn what is known by senior faculty and easy transference of knowledge of teaching within the community" (Pedagogical Patterns Project, 2007).

A special section of pedagogical patterns are e-learning patterns which many researchers have drawn attention to recently: "Researchers and practitioners in the e-learning field are attracted by the potential of design patterns, as a means to facilitate the capturing and sharing of different aspects of e-learning design expertise, to represent successful learning design models and to provide a lingua franca for joint course design" (Dimitriadis, Goodyear & Retalis, 2009).

Goodyear (2005) summarizes the meaning of the pattern approach for the field of education:
"…this pattern based approach has a good deal to offer to educational design, particularly in relation to:

- Providing the teacher-designer with a comprehensive set of design ideas
- Providing these design ideas in a structured way - so that relations between design components (design patterns) are easy to understand
- Combining a clear articulation of a design problem and a design solution, and offering a rationale which bridges between pedagogical philosophy, research based evidence and experiential knowledge of design
- Encoding this knowledge in such a way that it supports an iterative, fluid, process of design, extending over hours or days. "

Neumann, Derntl & Oberhuemer (2011) link the pattern concept to teaching methods and highlight the problem-oriented approach: "A pattern format would thus not be prescriptive (offering precise steps to be taken) about the teaching method but would provide an abstracted, problem-oriented presentation of a teaching method. Other formats would place different emphases including prescriptive steps of action to be taken as part of the teaching method, the teaching method's roots in educational or learning theory, or its value in certain subjects to foster understanding of typical concepts and processes". Wedekind (2011), too, links patterns to teaching methods.

Pedagogical patterns are often structures of activities and relations between learners, activities, and outcomes: "We define a pedagogical pattern as a teaching-learning activity sequence that is designed to lead to a specific learning outcome. Often we come across things called 'learning patterns' that are not specifically oriented towards a learning outcome; they are oriented towards a more general solution of a broader educational problem or goal" (Laurillard & Ljubojevic, 2011).

For e-learning, we can say that "E-Learning patterns capture good and successful practices and forms in educational contexts that make use of digital media. In describing such structures, we refer to 'design patterns' if the structures are constitutive, describing a design space, and analyze the forms in the general dimensions context, problem field and solution. An e-learning pattern captures the regularities of good practices in order to reuse the proven methods, scenarios and content forms in new contexts addressing new design tasks. The core idea is to not reinvent the wheel but to preserve what has been successful in the past. Beside the explicit description of good (successful) pedagogical methods, tools, media formats, resources, and scenarios design patterns reason about the adequate use of such solutions" (Kohls & Wedekind, 2011b).

Baggetun (2004) points out that "E-learning patterns are different from OOP patterns. E-learning patterns concern how technology mediates learning that is based on social interaction. This is completely different from the work of OOP patterns and even HCI patterns, which also are a more narrow interpretation of Alexander's original ideas." The E-LEN network highlights that patterns can capture educational values as well:
"The E-LEN network believes that e-learning patterns can and should be used to express educational values. We believe that e-learning should enable the enrichment of the learning paradigm in order to:
(a) support open, flexible and learner-centred patterns of study;
(b) provide new ways for learners to work collaboratively;
(c) facilitate the development of communication and co-ordination skills, and
(d) encourage the development of technological skills" (E-LEN brochure).

As we have seen there are many different views on patterns. This challenge has been long known in the pattern community (Corfman, 1998). To shed light on the similarities and differences we will discuss the key properties and perspectives in the next sections using the path metaphor introduced before.

2.3 The structure of patterns[3]

We will first discuss the basic components of patterns, i.e. the context, problem, forces, solution, and consequences, pattern languages and the relations between those concepts. Each concept will be illustrated by a physical path in a landscape – which is a metaphor for a structural configuration that fits into an environment – followed by examples from software and educational design. A discussion for each subsection will support the claims expressed in the illustrative metaphor and link the concepts to their roots.

2.3.1 Understanding context

2.3.1.1 Environments: Contexts in landscapes

A context is the environment or the situation in which we find ourselves in. In the simple example of a hike, the context is the physical environment of the landscape as well as your current situation (where you are, how well trained you are, what your hiking skills are, which tools you have at hand etc.) and your goal or intent. Besides reaching your destination, your intention may be to find ways that are not too tiring or to find ways that are physically challenging. You may prefer a path along scenic views or cultural attractions and make this a requirement for the solution.

[3] An earlier version of chapter 2.3 has been published as "The Structure of Patterns" at PLoP 2010 (Kohls, 2010b).

The context depends to some extent on your personal intent and preferences while at the same time the environment – the hills, canyons, slopes, lakes, cabins - is out of your hands.

At first glance, the context of that landscape seems to be fixed and static. However, the context can vary. One can hike at different times of the year; there will be different animals that cross the path, and there may be different groups of people who take the same trail. Hence, there is a certain degree of variability in the context. The power of a pattern is that it is open enough to be adapted to specific needs of an actual situation. While the core matter of a general context – in our example the shape of the landscape - remains invariant, we must not forget that there is an infinite number of ways in which the "same" context will change dynamically.

If the differences of a context are significant then another pattern might provide a better solution. The form of a solution is only adequate and of value if it still fits the context. For example, the very same form is no longer a solution if an essential part of the context (such as our starting position) changes.

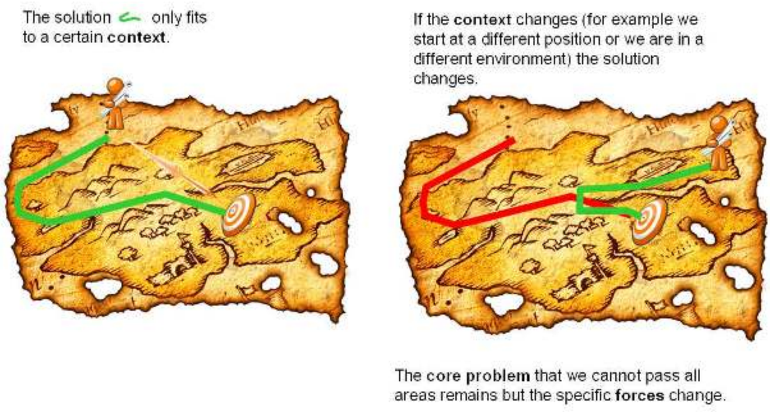

The solution only fits to a certain **context**.
If the **context** changes (for example we start at a different position or we are in a different environment) the solution changes.
The **core problem** that we cannot pass all areas remains but the specific **forces** change.

Figure 4. The solution form has to fit to the context.

A change in context can make new paths possible as well (e.g. a bridge over a canyon) or eliminate paths that have worked in the past (e.g. a rock slide blocks the way). The context implies which solutions could work because it contains the forces and problems that need to be resolved by the solution.

2.3.1.2 Examples for contexts

Examples for contexts in the development of software design are system size such as "a large system that requires decomposition" (LAYERS in "POSA-1" book, Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996), design situations such as "the algorithms for creating a complex object should be independent of the parts that make up the object and how they are assembled" (BUILDER in "Gang-of-Four" book, Gamma, Helm, Johnson & Vlissides, 1995), system flexibility, decoupling components to make parts of a system more independent (such as PROTOTYPE, or PROXY), representation of data and algorithms (such as COMPOSITE to represent part-whole hierarchies of objects, or STRATEGY for flexible change of algorithms), constraints when a pattern should not be applied (the INTERPRETER is only adequate for simple grammars and situations where efficiency is not critical), concrete tasks such as processing data streams (PIPES AND FILTERS), application types and application domains (security patterns, see Schumacher, Fernandez-Buglioni, Hybertson & Buschmann, 2006; fault-tolerant software, see Hanmer, 2007), design methodologies such as eXtreme programming, SCRUM or agile software design.

Examples for educational contexts are organizational forms (school, university, professional training), knowledge domains (science, art, economics), political issues (budgets, funding, required evaluation), skills of teachers and students, complexity of the content, teaching goals, student's goals, number of participants, schedules, amount of time, attitudes, available equipment and resources, pedagogical history and traditions etc. Patterns such as SEMINAR, LECTURE, TRAINING, PROJECT WORK, WRITING A THESIS are the context for other patterns such as BRAINSTORMING or PROJECT TIMELINE[4]. The context provides guidance for the appropriate application. If students are asked to write their thesis right at the beginning of their study, this assessment would not fit at all. But once students have gained some basic knowledge of their domain and have written their first texts as part of projects, they may be fitter for this task – or the task may be fitter to the skills of the students. One can put it either way. It is at the end of a study that students are best prepared to write a sophisticated thesis.

[4] Note that these patterns certainly exist in educational practice. However, they have not been documented in the pattern format yet. Nevertheless I will use capital letters to indicate that they are patterns in the world.

Likewise, brainstorming activities fit better to creative tasks such as PROJECT WORK. A brainstorming does not help much in solving mathematical equations.

2.3.1.3 Discussion of fitness between form and context

Our only way to describe a form not in isolation is to include a description of the context(s) to which it does fit, to which it belongs. Simply describing a method, tool, scenario, setting or media format is almost pointless because the meaning and the value of such forms fully depends on the context. A method is only of value if it is applied in the appropriate context. Thus, we cannot say that "creating a mind map" is generally a good thing because it is only meaningful if it serves the task at hand.

The context describes the situations in which the pattern can help – its applicability: "What are the situations in which the design pattern can be applied? What are examples of poor design that the pattern can address? How can you recognize the situations?" (Gamma, Helm, Johnson & Vlissides, 1995).

A precise description of the applicable contexts helps to avoid using the pattern in the wrong situation: "The context plays an important role in a pattern, however, since it defines the situation to which it applies. The more precisely this situation is described, the less likely it is that developers will use the pattern inappropriately" (Buschmann, Henney, & Schmidt, 2007, p. 42).

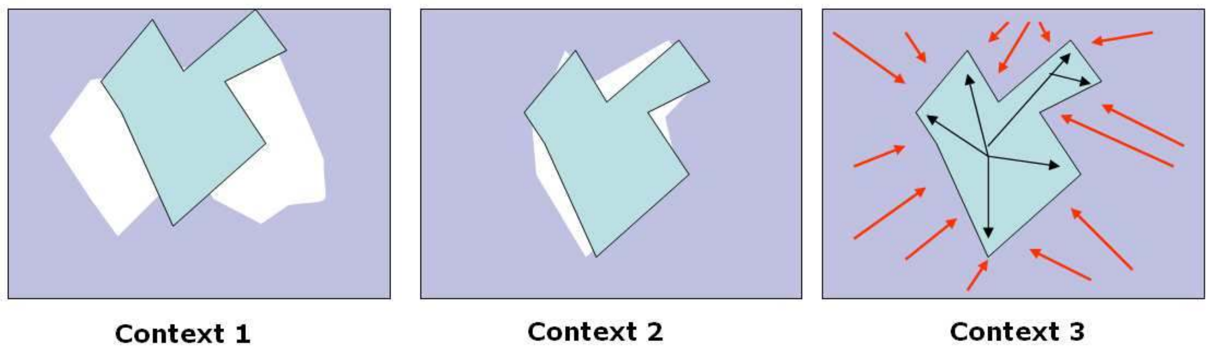


Figure 5. Various degrees of fitness between solution and context.

In figure 5 the very same form is used in different contexts. It does not fit to context 1, only roughly to context 2 and perfectly to context 3. In the last case we have a full fitness between the form and its context. Only then becomes the form a solution. Since a form becomes a solution if and only if it fits to the problem in the context, it is necessary to describe the context as well: "The form is a part of the world over which we have control, and which we decide to shape while leaving the rest of the world as it is. The context is that part of the world which puts demands on this form; anything in the world that makes demands of the form is context. Fitness is a relation of mutual acceptability between these two" (Alexander, 1964, p 19).

In a simple 2D-environment (figure 5), when the context and the solution fit, they have at their interface the same form. The form of the context, hence, provides requirements how the solution has to look like. Real world contexts are multidimensional, and the dimensions are semantic rather than in physical space. In the last picture, the arrows indicate forces that are shaping the form of the context. These forces "force" us to choose a specific form as a solution: "the unitary description of the context is […] also a description of the required form" (Alexander, 1964, p.21). Each situation consists of a field of forces that are interrelated. If these forces are not balanced we do have a problem. In order to understand the forces we must understand the context because this is where they arise: "If the context is not part of the pattern, then neither are the forces. However, if the forces are part [of] the pattern – which they must be, otherwise patterns just become solutions – they must be rooted in the context and have arisen from there" (Buschmann, Henney, & Schmidt, 2007, p. 203).

Because the context contributes to what a form is, to the meaning of the form, we cannot reason about it without referring to it. "To make the pattern really useful, we must define the exact range of contexts where the stated problem occurs, and where this particular solution to the problem is appropriate" (Alexander, 1979, p. 253).

2.3.1.4 Context connects patterns

Very often a situation is the result of previously applied patterns. For example, running an ONLINE TRAINING (see appendix F) puts you in a new situation. Patterns elaborate on this new situation in "Resulting Context" or "Consequences" sections. The new situation suggests further applications of patterns on a lower level. Vice versa these more specific patterns can be applied in the context of the higher level pattern such as ONLINE TRAINING. The context "describes the space in which the pattern can be embedded. It often includes references to patterns at a higher level of granularity that have been considered before" (Schümmer & Lukosch, 2007).

If we have a rich vocabulary of patterns then we can describe the context in terms of those patterns. The context description becomes dense. "If a pattern is applied in the context of another pattern, as it is in a pattern sequence, then we can narrow the description of the context to define it as that preceding pattern" (Buschmann, Henney & Schmidt, 2007, p. 204).

2.3.1.5 Context describes situations

However, the context is more than a set of previously applied patterns. While the pattern history contributes to the current situation, other factors are important as well: "Context includes a history of patterns that have been applied before the current pattern was considered. It also specifies size, scope, market, programming language, or anything else that, if changed would invalidate the pattern" (Coplien, 1996, p.8). Context is determined by the choice of previous patterns (such as running an ONLINE TRAINING), the general domain (such as e-Learning), and specifics of a new situation.

This has lead to some misconceptions about what the context description is technically. Borchers (2001) claims about context: "[context] turns a loose collection of patterns into a pattern language", "context and references make a pattern language" (p.71). In my view, it is not correct that the contexts alone make a pattern language. Rather, a context description can refer to other patterns of a pattern language and explicate the relations – even their correct combinations as in a grammar. But it "is not simply glue: there is the implication of fit, not simply adhesion, between context and problem" (Buschmann, Henney & Schmidt, 2007, 2007, p. 46). The context is not just the "'inverse function' of the references" as claimed by Borchers. If someone buys a company, the resulting context might be that a lawyer is required (BUY A COMPANY refers to CONSULT A LAWYER). However, the contexts in which a lawyer can be helpful are much broader than buying a company. Therefore, the context of the pattern CONSULT A LAWYER would not only point back to BUY A COMPANY but to many other situations as well; it could even be that it does not refer to BUY A COMPANY at all. There is an asymmetry between the relations. Buying a company requires a lawyer most of the times. But maybe only a small percentage of all lawyer consultations are about buying companies.

There is another reason why BUY A COMPANY might not be included in the explicit context descriptions of CONSULT A LAWYER. Rather than listing all the particular cases in which a pattern could be used, it should describe the general context of application: "To provide a general context statement we aim to typify the kind of context by its common features – an intensional approach – rather than by defining the context class by listing its members – an extensional approach" (Buschmann, Henney & Schmidt, 2007).

Describing the exact range of applicable contexts is empirically impossible. There could always be more situations where the pattern can help to solve a problem. Likewise, there could be certain circumstances in which it fails. If we know from experience that a path works in the summer, we do not know whether it works at other times of the year. To say that it works only in summer could be wrong because it might be fine in spring and autumn as well. To say that it generally works could be wrong because in the winter snow may block its passage. So, rather than referring to summer we could describe the conditions under which the path worked. If those conditions are found at other times of the year, it is likely that the path works as well. While the context should be open it should not be expressed to vague: "An overly general context that acts as an umbrella for many possible problem situations may be too vague, leading to inappropriate applications of a pattern. On one hand, too much detail will probably yield an unreadable pattern prefixed by an exhaustive 'laundry list' of specific context statements. On the other, an overly restrictive context may prevent developers from applying a pattern in other situations in which it is also applicable" (Buschmann, Henney & Schmidt, 2007, p. 44).

2.3.2 Understanding problem and forces

2.3.2.1 Problem and forces in a landscape

In a changed context the same solution form may not fit any longer. Another context sets new constraints, boundaries, introduces new requirements and offers different opportunities. For example, the equipment and training of a hiker is part of the context and implies different fields of forces. An untrained hiker might be forced to choose a path that is not too steep. The right equipment might offer new opportunities (e.g. climb a steep wall) but also introduce new limits. Constraints, boundaries, requirements and opportunities are different types of forces.

Forces can support each other or conflict if they imply different approaches. Such conflicts are problems that need to be resolved. The problem and forces of a given context are usually discussed in separate sections because they are important to understand why a pattern is shaped the way it is. If forces are the reason to build a solution in a specific way that works, we can consider forces as the cause that leads to the form of a pattern.

The first force we encounter is the one that drives us to our new destination. Our motivation (i.e. our desire to move on) is based on the fact that the current situation is less favourable (i.e. problematic) than the intended situation. The next problem we encounter is that we cannot go straight to our destination. The rock face blocks our way. It **forces** us to take another way. That force is in conflict with our desire (or "force") to reach the target directly.

If we are looking for a way that brings us to our destination we have to account for **all the forces** that influence our chosen path in the given context. There are some forces that cause the original problem and there are more forces that shape the path of the solution.

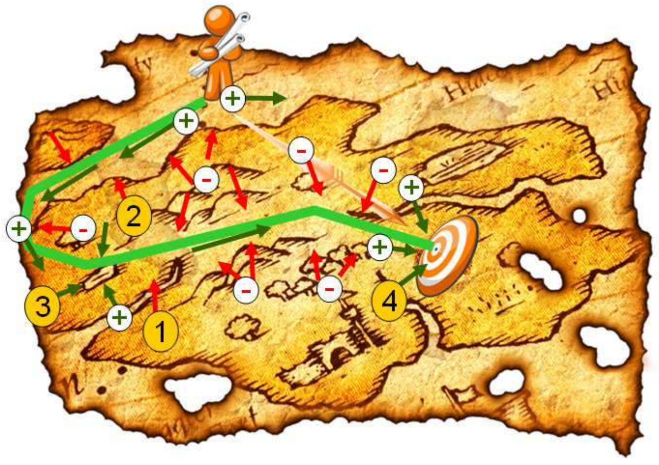


Figure 6. Positive and negative forces.
The (-) arrows denote forces that deny us a specific path, e.g. we cannot cross over a canyon (1) or climb a steep wall (2). The positive forces (+), on the other hand, are attractive ways to go. There are also some attractive places to which positive forces point, such as a place to get fresh water (3), or the actual destination (4).

In order to capture the forces we need to analyze the relation between the solution and the context. A key is to ask why questions. Why do we have to follow this specific path? Why do we have to go there? Why do we need this specific form? Why can't we do another thing instead? Each single force gives another answer to such why-questions. A force explains the cause for a specific design decision by giving the "because" to the "why". In explaining the reasons for the solution form it helps us to understand the pattern rather than blindly following an advice. This understanding of a pattern is essential for judging whether it fits to the problem at hand and to adapt it to the specifics of the situation

2.3.2.2 Examples

Many forces in software design are due to the decomposition of a system into objects, such as the need for "encapsulation, granularity, dependency, flexibility, performance, evolution, reusability, and on and on. They all influence the decomposition, often in conflicting was" (Gamma, Helm, Johnson & Vlissides, 1995, p. 11). Other factors are performance, efficient handling of resources, safety, transparency, maintainability, robustness, consistency, reduction of complexity, avoiding overheads, exchangeability of components or algorithms, re-use, adaption between different systems (ADAPTOR, BRIDGE, MICROKERNEL; Gamma, Helm, Johnson & Vlissides, 1995; Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996), dynamic changes of data (UNITS OF MITIGATION, Hanmer, 2007).

Conflicting forces in education include "the contradiction between the topics to be taught according to the curriculum on the one hand, and the fields of interest of the students on the other hand" (VIDEOCLIP, Kortenkamp & Blessing, 2011), students underestimating time for projects (patterns for supervising thesis projects; Schümmer & Schmolitzky, 2009), cognitive limits, concentration, handling many students, splitting time between research and teaching, personal conflicts, struggling with too many lessons, balance between freedom and guidance, skills need to be checked, pressure before exams, fair ratings, cheating, or time constraints.

2.3.2.3 Discussion of problem and forces

To balance the forces means to resolve conflicts and problems wholesome, leading to a state of harmony: "A man is alive when he is wholehearted, true to himself, true to his own inner forces, and able to act freely according to the nature of the situations he is in" (Alexander, 1979, p.105).

If the forces of a situation are unbalanced, a problem arises. The problem is a conflict of forces that are rooted in a context: "**we can express the problem as a conflict among forces** which really do occur within the stated context, and cannot normally be resolved within that context" (Alexander, 1979, p.238; highlighted in original). We feel uncomfortable if our inner forces are in conflict with the outer forces, i.e. if we are not in harmony with our environment. This lets us recognize a problem – something is wrong.

However, the field of forces is complex; very often we are not aware of all the forces that are at work. As a result we may perceive that something is wrong - that there is a problematic situation – but we do not understand its nature. "The difficulty is that we have no reliable way of knowing just exactly what the forces in a situation are" (Alexander, 1979, p. 285) whereas experts „know how to play with the various forces in their area of expertise". (Schümmer & Lukosch, 2007, p. 24).

The discussion of forces is very important to better understand the problem. Forces help to realize why we have to intervene and why to do it in a specific way. "If we understand the forces in a pattern, then we understand the problem (because we understand the trade-offs) and the solution (because we know how it balances the forces), and we can intuit much of the rationale. For this reason, forces are the focus of a pattern" (Coplien, 1996). Forces are the very reason for a pattern. Forces provide the rationale for the pattern's structure. In understanding the forces, we are better equipped to perform adoptions to specific situations and to avoid misapplications: "Forces are also the key to understanding why the problem's solution is as it is, as opposed to something else. Finally, forces help prevent misapplications of a pattern in situations for which it s not practical" (Buschmann, Henney & Schmidt, 2007, p.37).

2.3.2.4 The core problem statement – conflicting forces

Not all forces acting in a context are necessarily the cause of a problem. Usually there are just a few forces that are troublemakers and force us to do something about the situation. The problem statement usually captures the forces that are actually problematic. It should tell the reader what is wrong in a specific situation and state a serious problem that makes it obvious that we have to do something about it: "And there is an imperative aspect to the pattern. The pattern solves a problem. […] In this sense, the pattern not only tells him how to create the pattern […]; it also tells him that it is essential for him to do so, in certain particular contexts, and that he must create this pattern there" (Alexander, 1979, p.183).

In an educational setting, a text or an illustration that is too complicated or assumes knowledge of students that they do not have very often is the cause of trouble. The problem is that students will not understand the resource and this will cause further problems, e.g. it becomes harder to follow what comes next or students may stop concentrating on the content altogether. The problematic force is that learning resources have to be suited to the needs and levels of skills of the students.

Forces are not to be mistaken only with physical forces as the previous example has shown. Patterns move past "technical and mechanical forces and take human forces into account" (Coplien, 1996). Forces can also be seen as human needs or interests (Borchers, 2001).

The importance of explicitly naming the problem and illustrating what is problematic and which negative consequence the problem has cannot be underestimated. Very often people think patterns are about providing neat solutions to a problem one has. But the true power of patterns is that they are capable of illuminating problems one has not been aware of. Taking the poor design of complex and overloaded illustrations we can often find we see that many educators and docents are not aware of the problems caused by badly designed visualization. They do not see that a picture overloaded with information is a problem for cognitive processing.

Here, problem means to understand that there is a conflict between a form and its environment.

2.3.2.5 The problem of balancing forces

Once we tackle the initial problem, we have to account for all the forces of the given situation, not only the ones that originally caused the problem. The reason is that we are not dealing with a "set of forces" but a "field of forces". That means that whenever we do something about one force, this intervention will influence other forces as well. The problem of design is then, to find a configuration that does not raise conflicts to other forces: "we must define the problem, or the field of forces which this pattern brings into balance" (Alexander, 1979, p. 251).

For example, once we address the problem that learning resources should be suited to the needs and skills of each individual, we will encounter forces that have not been part of our core problem. There are forces that come into play when we try to assess the current level of skills, such as not creating a situation in which the students feel uncomfortable by being tested. When we start to individualize the resources, the force that we have not enough time to create an individual worksheet for each student by hand becomes relevant. Of course, the time of a docent has been limited all along (the force has been in the context already) but it is only in the process of finding an appropriate solution for individualization that we fully experience this limitation.

This new raise of conflict when we think about potential interventions is often expressed explicitly: "Other prose styles emphasize the conflict between different forces, presenting first one assumption or desired outcome, and then contrasting it with another that is in conflict, often introduced with a 'but' or 'however'" (Buschmann, Henney & Schmidt, 2007, p. 99).

It is therefore, that a balance of forces leads to the solution form. In *Notes on the Synthesis of Form*, Alexander refers to D'Arcy Thompson in explaining forces: "D'Arcy Thompson has even called form the 'diagram of force' for the irregularities" (Alexander, 1964, p. 15).

The problem of balancing all the forces is even more serious than the core problem because we have only limited understanding of this field: "What does make design a problem in real world cases is that we are trying to make a diagram for forces whose field we do not understand" (Alexander, 1964, p. 21).

Here, problem means to understand the field of forces correctly in order to find a balancing form – the solution.

2.3.2.6 The problem of implementation

Let us assume that we have found a solution form that balances the forces to a satisfying degree. For example, a specific e-assessment form may provide appropriate ways to find out the current levels of skills, allowing the teacher to adapt the pace of the lesson, provide additional examples if needed or rephrase a concept. Such a form is certainly a solution to the problem in question. However, how to build or implement that system is a different problem altogether. It is one thing to know how a final product can help to serve in a specific context. But knowing how to build that thing is a problem on a different level. In the same way as there may be many solutions to the same problem, there are many ways to build a particular solution. Very often for the majority of users there is no need to create their own system. They can simply use an existing e-learning system or program. Of course that does not resolve the problem of implementing such a system but not everybody has to take care of that problem.

The structure of unfolding a solution (the "process") is different to the solution itself (the "thing"). Forms unfold according to their forces; this idea is again borrowed by D'Arcy Thompson (Gabriel, 1998): "Alexander says: 'What Thompson insisted on was that every form is basically the end result of a certain growth process. When I first read this I felt that of course the form in a purely static sense is equilibrating certain forces and that you could say that it was even the product of those forces – in a non-temporal, non dynamic sense, as in the case of a raindrop, for example, which in the right here and now is in equilibrium with the air flow around it, the force of gravity, its velocity, and so forth – but that you did not really have to be interested in how it actually got made. Thompson however was saying that everything is the way it is today because it is the result of a certain history -

which of course includes how it got made. But at the time I read this I did not really understand it very well; whereas I now realize he is completely right'" (Gabriel, 1998).

Here, the problem is to find the right sequence of steps to create the form that is in balance with its environment.

2.3.2.7 The various kinds of problems

We have seen that different kinds and levels of problems arise in specific situations:

- The **core problem** describes what is problematic in that situation, i.e. there is something we need to change because it has negative effects.
- The **problem of finding a solution** is to look for a fitting form that balances all the forces that act in that situation, not only the problematic ones. Since all forces are interrelated this is not a trivial task.
- The **problem of building the solution** tackles the question of how to generate the form of the solution.

The core problem and the forces are usually explicitly captured in a pattern description. Some pattern formats use different sub-headlines for the core problem (the problematic forces) and the discussion of forces in general. The Alexandrian format only knows a section in which the forces are discussed but the core problem is highlighted in bold. Either way a discussion of forces is considered as a basic component of a pattern description.

The problem of building or creating is usually not further discussed as a problem but directly tackled in the solution. This is usually done in a section "implementation". However, some pattern descriptions do not address the problem of building at all.

Unfortunately, there are also pattern descriptions that only address the problem of "how-to" and do not discuss the core problem. Every now and then we find pattern descriptions that have a problem statement that looks somewhat similar to this:

- How to setup a wiki?
- How to find good questions for your tests?
- How to do X?

These "how-tos" do address a problem, i.e. the problem of implementation or building the solution. This is indeed usually a difficult problem that needs to be addressed. However, this problem only arises once we have already decided which solution to use. These are problems of a different level and there is a consensus in the pattern community that a pattern description should contain the original core problem and the forces that influence the form of the solution.

The discussion of the core problem gives answer to the question why we need a solution. Why do we need a SHARED GROUP SPACE, a DISCUSSION FORUM, a BLOG FARM, an E-PORTFOLIO in the first place? The discussion of the forces gives answer to the question why the solution does have the form it has. It is particularly in the forces that the requirements and constraints for the solution are explained. The reasoning about a pattern is found in the discussion of the forces. Since the forces explain the solution, their discussion helps readers to understand the solution. This is critical to adapt the solution to specific contexts. The how-to is usually directly addressed in the solution description and not raised as an explicit question.

2.3.3 Understanding solutions

2.3.3.1 Paths as recurrent solutions in landscapes

A solution is one known way that takes into account all the forces that matter in a specific context and balances them to a satisfying extent. The form of the solution is a path that has proven in the past to actually lead to the intended goal and takes the forces appropriately into account. It does not mean that it is the only path that does exist nor does it claim to be the best path. There might be paths that even fit better to the context but that have not been found yet.

The term **pattern** suggests recurrent solution forms. And that is what happens if people use a path: no two hikes along a path will be identical. We will use detours, walk crisscross and unfold the hike every time in new a way while still following the same way. That refers to Alexander's second element of the definition of patterns. It is a solution *which occurs over and over … without ever doing it the same way twice*.

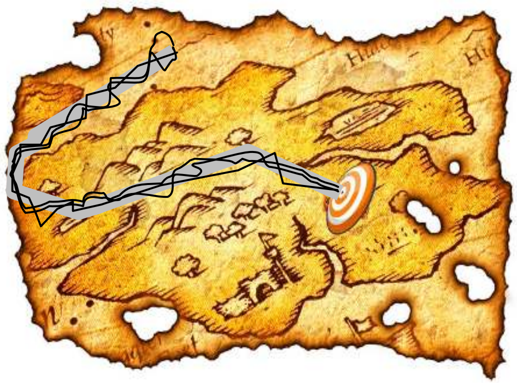


Figure 7. Each hike on the same path is different yet the same.

While there is variance, the general structure is preserved. Every concrete journey along a path has the same structural quality even if each instance unfolds differently. This structural symmetry lets us find an abstracted representation of an actual path. However, the range of possible variation is different for the various parts of a path. At some points the space of a path may be broad, allowing many different steps to move along the same path. At other points, the path may be narrowed or obstacles only allow specific moves. Those points are critical and need more attention to make the path a success. It is therefore that we must find a medium level of abstraction. Like a map, a pattern description has to provide enough information to actually follow the solution path. It will leave out irrelevant details and concentrate on hallmarks that help designers to find their way. But it must not be too abstract. Otherwise we are loosing the form and the specific structure that guides the designer through the territory.

2.3.3.2 Examples

Patterns in software design illustrate the object structure using class diagrams (OMT or UML), the dynamics or interactions at run time, implementation issues, examples and sample code, sometimes Class-Responsibility-Collaboration cards (Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996) and a discussion of variants.

Patterns in education sometimes illustrate typical situations in a photograph or a sketch; other patterns illustrate the sequence of activities in a formal diagram (Derntl, 2006) or an informal didactical map (Döbeli-Honegger & Notari, 2011). Some patterns have a separate section for obstacles (e-teaching.org, see Kohls, 2009b). It is also common to discuss the preparation, the delivery and the wrap-up separately. Many patterns suggest the support of tools (Schmolitzky & Schümmer, 2008; e-teaching.org).

2.3.3.3 Discussion of solution

A solution is a form or structural configuration that balances the forces of the context in order to resolve the tension between the problematic forces: “We see, in summary, that every pattern we define must be formulated in the form of a rule which establishes a relationship between a context, a system of forces which arises in that context, and a configuration which allows these forces to resolve themselves in that context. It has the following generic form: Context → System of forces → Configuration” (Alexander, 1979, p. 253).

It is important to notice that a good solution has an inner harmony (no conflict of forces in the structure of the solution) and an outer harmony (no conflict of forces with the structure of the environment). The very same form can or cannot be a solution for the problem at hand. Whether a form is appropriate for the given context depends on the structural form of the context and the forces found in that context because “a pattern only works, fully, when it deals with all the forces that are actually present in the situation” (Alexander, 1979, p. 285).

2.3.3.4 Solutions balance the field of forces

A form that does not balance the field of forces is not a whole and true-hearted solution: “For this reason, we favor characterizing successful solutions that address the forces in a problem adequately as ‘whole’ and those that do not as ‘dysfunctional’. The former term entails balance and completeness – whole as opposed to half-done and half-baked – whereas the latter suggests that a design is impaired in some way and may undermine the stability of a larger whole” (Buschmann, Henney & Schmidt, 2007, p. 41).

If the forces in a system – technical requirements as well as human desires - are adequately resolved, the "Quality without a Name" (Alexander, 1979) emerges. Such a solution is a stable pattern, it is "utilitarian, aesthetically pleasing, robust, and whole" (Coplien, 1998e). Good form is not just defined functionally but aesthetically as well (Coplien, 1998b).

A pattern provides a general solution in the sense that there are many different ways of instancing the solution form. A pattern defines a "coherent yet infinite design space" rather than single realizations of a solution (Buschmann, Henney & Schmidt, 2007, p. 76). It is general enough to not prescribe every step of an actual implementation. Since every particular situation is different - thus imposing a unique configuration of forces - there must be open space to unfold a general solution according to the needs of the particular field of forces.

Yet it is specific in that it shows the limits of the design space of varying solutions that appropriately address problems of the same kind. It builds on the lessons learnt, preventing the designer to run into typical pitfalls, and it draws from successful approaches of the past: "A good solution has enough detail so the designer knows what to do, but it is general enough to address a broad context" (Coplien, 1996).

2.3.3.5 Sub sections of solution descriptions

The solution of a pattern is usually described in several sub sections of a pattern document, each investigating another aspect of the solution. Very often, the solution starts with an outline of the core of the solution. For example, we could explain in a few words what a MULTIPLE CHOICE TEST is, giving the reader a brief overview. The outline of the solution often directly mirrors the needs of the context, i.e. it is a necessary form due to the context. There are many pattern formats that start the description of the core solution with a "Therefore…" indicating that the form of the solution is a consequence of the context and its forces. While we can give a brief statement about most forms, we could also provide more detailed descriptions about the form, in particular about its inner structure and how the parts of the solution relate. For software design patterns we frequently find a section entitled "interaction" which discusses in more detail how the elements of the solution work together to form the solution. For a MULTIPLE CHOICE TEST we could discuss the single elements, e.g. how to formulate a clear question and how distracting answers should look like.

2.3.3.6 Implementation

In the same way as the problem consist of multiple levels (i.e. problematic forces, the problem of balancing all forces in the context, and the problem of creating the form) the solution can be discussed at various levels. Most importantly we can show how to build the solution: "The solution part of a good pattern describes both a process and a thing: the 'thing' is created by the 'process'" (Buschmann, Henney & Schmidt, 2007, p.33).

Software patterns very often have a section "implementation". For the pattern MULTIPLE CHOICE TEST, the pattern description could explain how to create meaningful MULTIPLE CHOICE TESTS by describing useful tools and a sequence of activities. It is also helpful to explicitly point out obstacles, i.e. one can show which parts of the implementation need specific attention because if not mastered with care the whole implementation could be a disappointment. There may be some aspects of the solution where it does not matter how it is done exactly whereas other parts of the implementation may require a very precise performance by the textbook.

We have seen that recurrent parts of the solution form can be outsourced into separate patterns. Likewise recurrent actions, use of tools, or handling of obstacles can be extracted into separate pattern descriptions that are only referred to in the implementation section.

2.3.3.7 Different ways to achieve a solution

If there are different patterns that can solve the same problem, there are also different ways of achieving a specific pattern. The sequence of steps in creating a solution is a pattern, too. And there can be different patterns of sequences that have the same solution as a result. For example, there are different ways to create a MULTIPLE CHOICE TEST, different ways to run an ONLINE TRAINING, different ways to organize an EDUCATIONAL BLOG. While the ways are different, the results are the same (the same form/pattern).

It is interesting to notice the difference between process patterns (e.g. methods, workflows etc.) and artefact patterns (e.g. tools, settings, media types). The form of a method is created in its performance, e.g. the steps of the method are the way of the method. Hence, the form of a method is non-permanent. We can only trace its form by capturing the single steps of the performance. This sequence of steps in a method is the pattern itself, i.e.

the form of the sequence and the form of the solution pattern are identical. However, if we create a tool or an educational resource, such as an interactive graphic, the sequence of steps that create this artefact has a different form than the artefact that results from the process. In section 4.2.1.4 we will see how wizards can guide through the sequence of steps to build a configuration.

2.3.3.8 The role of examples and known uses

The actual pattern is the form of the solution and that is the reason why patterns most frequently are named after the solution. A pattern description does not limit itself to the description of the solution but explains the solution (in terms of the forces), describes the contexts in which the form becomes a solution, and shows how to create the solution.

However, all of these descriptive entities are only an indirect account of the solutions of the design space. The description of the solution and its details are analytical and take the solution apart by means of words. Of course such parts acting together can capture the meaning of the pattern as a whole - but only in indirect ways. Therefore, a pattern description usually contains examples that show instances of the pattern as a whole. The term "pattern description" is not quite precise because an example is not a description of the pattern but an instance of the pattern. The character of the pattern manifests in such an instance, the concept of the pattern is exemplified.

Yet there are different ways how examples are integrated. Examples can be referred to, described, represented, modeled or actually be included.

Very often the "Known uses" section only refers to examples. They provide support that the described pattern is indeed a recurrent form. Those examples are only listed or described briefly. Sometimes one case study is picked out for a detailed illustration.

Many pattern descriptions contain examples by showing pictures that are typical for the situation. A photograph of a GROUP DISCUSSION shows an example of a group that discusses. A screenshot of a MULTIPLE CHOICE TEST shows an example of a multiple choice test. These illustrations are representational. A screenshot of a MULTIPLE CHOICE TEST is not the thing itself just as a photograph of a dog is not the dog.

Descriptions and representations are only references to the actual example. But pattern descriptions can include actual examples or samples that exemplify properties of the pattern directly rather than by reference. If we include examples of good questions of a MULTIPLE CHOICE TEST, then these questions are real instances of MULTIPLE CHOICE QUESTIONS. If a software pattern includes code snippets, then we see example code and not only a representation or description of the code. Examples let us experience the pattern's solution by seeing or reading about an actual instance of the pattern.

2.3.4 Understanding consequences

2.3.4.1 Consequences of choosing a path

The context shown on the map allows more than one solution path. Each path has different advantages, disadvantages, liabilities, and suggests different next steps. For example, the path that leads to the historic site certainly has the advantage of visiting that place. However, it requires more time. The two alternative paths compete with each other.

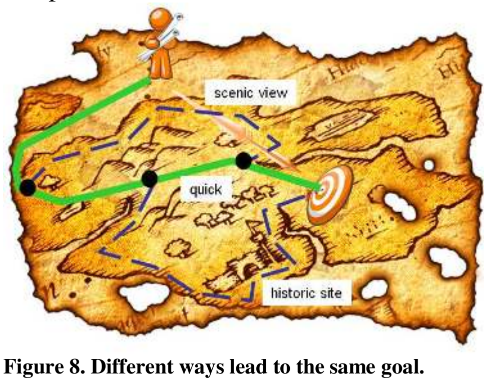


Figure 8. Different ways lead to the same goal.

In the process of finding a decision between different solutions, the pros and cons are weighed against each other; the consequences are compared. The overall situation will determine the actual path taken. If the hiker has a special interest in historic sites this interest "forces" him to take the longer way because he "has" to see it. In this context the path that runs along the historic site is best for that hiker. However, in a slightly different context there might be a time constraint to reach the goal. In that case the hiker might be forced to take the shorter path because reaching the goal in time (e.g. before sunset) might be more important than seeing the historic site. The better we understand the consequences the better we can find an appropriate decision. If we know the advantages (values) and disadvantages (costs) of a path we can better decide which one fits the problem at hand.

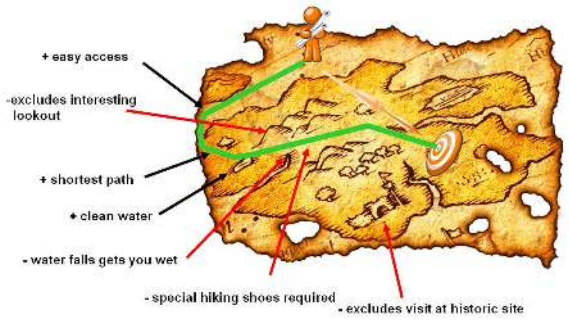

+ easy access
-excludes interesting lookout
+ shortest path
+ clean water
- water falls gets you wet
- special hiking shoes required
- excludes visit at historic site

Figure 9. Consequences of choosing one direction

Which consequences matter depends on the specific context or situation. Small changes in the context (time constraints, personal interests) have impact on the path we consider best. There are also situations in which two paths seem to be equally valuable. For example, if one path leads to a historic site and the other to a scenic view, and both are equally interesting for the hiker then the decisions might depend on mood or previous experience (if the hiker has seen already five historic sites he might choose a scenic view). Both are contextual properties as well. Unless one flips a coin each decision is based on the fact that one path is favoured over the other.

Once we have made our decision for one path we have to live with the resulting context and its liabilities. For example, we may encounter further obstacles on our way. The path may lead to our goal but we have to take care of some challenges. These challenges are local problems that occur in the context of the chosen solution. It is common to describe recurrent local problems and their solutions as separate patterns that can be used in the context of the current pattern. That is one reason why solutions frequently refer to other patterns.

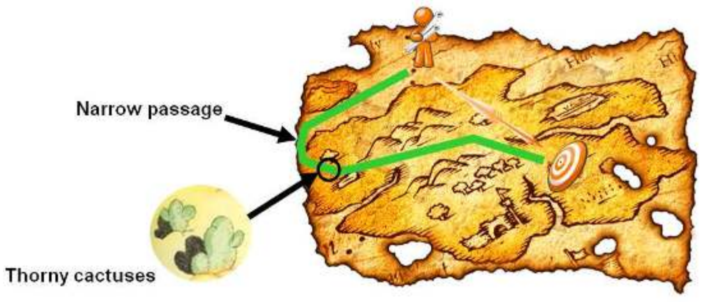


Figure 10. Obstacles on a path

The choice of a path also has consequences for the next steps. Once it is decided where to go there is a resulting context that suggests which sub-paths can be followed and which preceding and succeeding paths are available. The start and end point of a path have to connect to surrounding paths. At any time, the steps that can be followed next depend on previous decisions.

2.3.4.2 Examples

Consequences discuss the benefits and liabilities: "The consequences for software often concern space and time trade-offs. They may address language and implementation issues as well. Since reuse is often a factor in object-oriented design, the consequences of a pattern include its impact on a system's flexibility, extensibility, or portability. Listing these consequences explicitly helps you understand and evaluate them" (Gamma, Helm, Johnson & Vlissides, 1995, p. 3).

Examples for consequences in software design are reuse, support for standardization, dependencies are kept local, enhanced efficiency and lower cost, decoupling clients from the location of server components (LAYER), separation of housekeeping code from functionality (PROXY), decoupling interface and implementation, improving extensibility, hiding implementation details from clients (BRIDGE), limiting sub-classing, decoupling parts of a system, simplifying object protocols (MEDIATOR) etc. (these consequences can be found in Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996).

Typical positive consequences in education are fair feedback, motivation of students, deeper understanding of subjects, fostering creativity, or reduce of workload for students and teachers. Negative consequences could be long preparation times, variation in the degree of engagement for different students, dependence on technology, or higher financial costs.

2.3.4.3 Discussion of consequences

There is a strong relationship between consequences and the forces: "The consequences are the results and trade-offs of applying the pattern" (Gamma, Helm, Johnson & Vlissides, 1995, p. 3). This relates to the statement of the previous section that forces are the "trade-offs". We can say about the consequences similar things as about the forces: "By 'consequences' I mean the things that make the problem a problem and the solution a solution. In practice it is rare for solutions to be 'right' or 'wrong'. Instead, a proposed solution will typically do some things well and other things less well" (Koenig, 1998).

The consequences discuss the forces from a different point of view. They discuss which forces have been addressed (benefits), and which forces are not resolved (liabilities). In a perfect world all forces can be resolved; however, in the imperfect world software designers and teachers live in, we are faced with solutions that do not take all the forces into account (Buschmann, Henney & Schmidt, 2007). In such cases it is important to learn about the forces that are not addressed. In a particular design situation the forces may be not that relevant and we can live with the fact that they are not addressed. For example, if a teaching method does not address high talented students this is a general drawback of the method; however, if a class only has students of average talent this force needs not to be addressed.

2.3.4.4 Resulting context

Since the application of a pattern shifts us to a new situation we will also encounter new forces. To address the unresolved forces of the original context and the new forces of the resulting context a pattern can refer to further patterns that might resolve them. Patterns build on patterns:

"The Resulting Context is the wrap-up of the pattern. It tells us:

- which forces were resolved
- which new problems may arise because of this pattern
- what related patterns may come next

Each pattern is designed to transform a system in one context to a new context. The Resulting Context of one pattern is input to the patterns that follow. Contexts tie related patterns together into a pattern language" (Coplien, 1996).

2.3.4.5 Forces – before and after

While a forces section usually discusses the field of forces *before* the application of a pattern, the consequences or resulting context discuss the field of forces *after* applying the pattern. In that sense, forces are not only part of the context but also features and consequences of a solution (Rising, 1998). Consequences tell us what happens after an intervention, which effects are about to be expected. The consequences contribute to the understanding of what a pattern does to a problem or situation. We can evaluate the consequences and decide whether that is what we want in a particular situation.

Because the consequences discuss forces as well, it is an important key to decide which pattern is appropriate. An undesired consequence or effect could be seen as a constraint to a pattern's applicability: "The key to selecting the 'right' patterns is to consider the context, the forces, and the consequences of the competing patterns, and thus the context and the forces in the design situation itself" (Buschmann, Henney & Schmidt, 2007, p. 144).

Once we have resolved all of the forces by the interplay of patterns, we are reaching the ultimate consequence we are looking for: the quality without name. Some patterns may posses that quality to a certain extent. But usually it requires several patterns to accomplish it; that is why we need a pattern language to express a solution as a whole.

2.3.5 Understanding pattern languages

2.3.5.1 Connecting paths

Before we can follow a proven path we have to know that it exists. The more paths we know of, the more flexible we are to choose our routes. Once we have chosen a specific path to cross a mountain, we can label that path "Long Mountain Path". The name is already expressing some of its properties. We can now refer to it in other path descriptions that lead to it or follow up from there. In a description of the "Long Mountain Path" we could also refer to connected paths to provide further context. Instead of describing the "Long Mountain Path" step by step we could refer to sub-paths between intersections. This provides further flexibility because there may be alternative sub-paths between two intersections. Let us assume that we have seven intersections on our path and each intersection is connected by two different sub-paths. Then we can, by reference to the sub-paths, compose 2 x 2 x 2 x 2 x 2 x 2 = 2^6 = 64 different paths over the mountain. Instead of describing each of these paths (as different mountain paths) we only need to describe the twelve sub-paths and a more general (and abstract) "Long Mountain Path" that refers to the sub-paths. Of course we could further divide the sub-paths into smaller ones that take more alternatives into account.

The interconnections of patterns are captured in pattern languages. A pattern language consists of both patterns (the vocabulary) and rules how patterns can be used in conjunction (the grammar). Patterns build a vocabulary because each pattern description starts with a name that denotes the pattern. This is similar to referring to a path only by its name. We can express more things if we have a wider vocabulary. The more solutions we know, the easier we can generate more complex designs because we are able to express the designs in terms of those known solutions, i.e. the named patterns.

If we have a pattern language with a rich vocabulary we can make the relations between patterns explicit. Elements of the context and elements of the solution can be referred to by using pattern names rather than explicitly describing the structure. References to other patterns are highlighted in pattern description by using CAPITALIZED LETTERS. By those references, and sometimes in separate sections labelled "Related patterns", a grammar for the appropriate use and combination of patterns is partly defined. As part of a language those patterns can be used to generate infinite new instances by both combination and adaption: "A pattern language gives each person who uses it the power to create an infinite variety of new and unique buildings, just as his ordinary language gives him the power to create an infinite variety of sentences" (Alexander, 1979, p XI).

2.3.5.2 Examples

In *POSA-1* Buschmann, Meunier, Rohnert, Sommerlad & Stal (1996) speak of systems of patterns rather than pattern languages. A system of pattern shows the connections between patterns but does not describe a whole language. The Patterns for Fault Tolerant Software (Hanmer, 2007) are a very good example how patterns interweave to a language because each pattern refers to other patterns in the solution and context sections. Rather than listing related patterns, the pattern descriptions include references to other patterns in the text flow. Thus, the sub-patterns are deeply connected and contextualized. One of the best example for weaving patterns into a language can be found in Fearless Change (Rising and Manns, 2005) where each of the pattern depends on other patterns. An example for education is the SEMINARS pattern language (Völter & Fricke, 2001). We will also discuss a small language for Online Training in the empirical section of this thesis (see section 4.4 and appendix F).

2.3.5.3 Discussion of languages

Christopher Alexander considers pattern languages as "The Gate" to reach the "Quality without a Name". Before we discuss the elements and benefits of a pattern language, let us first consider Alexander's understanding of languages and pattern languages (Alexander, 79, p.183):

"From a mathematical point of view, the simplest kind of language is a system which contains two sets: 1. A set of elements, or symbols. 2. A set of rules for combining these symbols. The logical languages are an example. […] A natural language like English is a more complex system. […] A pattern language is a still more complex system of this kind. […] An ordinary language like English is a system which allows us to create an infinite variety of one-dimensional words, called sentences. […] A pattern language is a system which allows its users to create an infinite variety of those three dimensional combinations of patterns which we call buildings, gardens, towns. […] In summary: both ordinary languages and pattern languages are finite combinatory systems which allow us to create an infinite variety of unique combinations, appropriate to different circumstances, at will." The summary is supported by a table (Alexander, 1979, p. 187):

Natural language	*Pattern Language*
Words	Patterns
Rules of grammar and meaning which give connections	Patterns which specify connections between patterns
Sentences	Buildings and places

We shall now consider the three levels and their implications.

2.3.5.4 Patterns as vocabulary

In our natural language words are used to denote categories (Löbner, 2002). By "tree" we are referring to the category that includes all instances of trees. In the same way, a pattern name refers to a pattern, which means all the instances that manifest the pattern's structure.

Noble, Biddle & Tempero (2006) consider patterns as signs and discuss the semiotics of design patterns. Semiotics is the study of signs in society; a sign can be a two-part relationship between a signifier and a signified; for example "tree" is a signifier, and that what is a tree is the signified.

The pattern name "Observer" is a signifier that signifies the pattern description of the Observer pattern. The pattern description is itself a sign, a relationship between the pattern's solution (the actual form in a design) and its meaning (its intent and the problem solved).

The word "Brainstorming" signifies a (description of a) relationship where the activities during a brainstorming signify what that means, i.e. the intent or effects of a brainstorming.

Such a semiotic system of second order can explain the analogy of words in natural languages to patterns in pattern languages. A pattern name means the pattern description which denotes the actual pattern instances in the world and their meanings: "As an element of language, a pattern is an instruction which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces wherever the context makes it relevant" (Alexander, 1979, p. 247).

While the word "tree" means the trees of the world, the term "Looped Local Roads" means the pattern (or the pattern description) which means the actual instances of looped local roads (pattern instance).

As with natural language, the patterns of a pattern language can be used to express actual structures. And we can use words of natural language to refer to patterns. This has two consequences.

First, by giving names to patterns, we can refer to the patterns themselves and to the implementations of the patterns. We can mean both the concept of an OBSERVER and the actual OBSERVER implementation; we can mean both the concept of a BRAINSTORMING and an actual activity of brainstorming. This gives designers an important vocabulary for communication: "Their names collectively form a vocabulary that helps developers communicate better" (Vlissides, 1997). It offers an opportunity to express what a design team wants. This improves the communication between different stakeholders, such as developers and clients, or members of an interdisciplinary team. A pattern language can "create the vocabulary for a lingua franca, a common language, between designers and users" (Borchers, 2001, p. 29).

It also makes it easier to document a system because we can refer to complex structures by simply using the name. "Having a common vocabulary means you don't have to describe the whole design pattern; you can just name it and expect your reader to know it. A reader who doesn't know it will have to look them up at first, but that's still easier than reverse-engineering" (Gamma, Helm, Johnson & Vlissides, 1995, p. 352).

The second feature of having a large vocabulary of patterns is even more important. The more solution one knows, the better one can generate whole systems. Without knowing the solutions one cannot use them. "If a man has a great deal of experience of building houses, his language for houses is rich and complex; if he is a greenhorn, his language is naive and simple. A poet of houses, a master builder, couldn't possibly work without his language – it would be as if he were a greenhorn" (Alexander, 1979, p. 207). Without knowing the BRAINSTORMING method, one cannot apply it. We all have pattern languages in our mind (Alexander, 1979, p. 202). Making this language explicit allows us to share our knowledge, discuss it and enrich our vocabulary.

2.3.5.5 Connections between patterns

The second element of language is the set of rules for combining the symbols. In natural language these rules are the grammar and meaningful statements – syntax and semantic.

In a pattern language the patterns themselves introduce connections to other patterns. This is done by "related patterns" sections or referring to other patterns by their names (highlighted by capitalization). If the context of a pattern refers to other patterns, this indicates that its application makes sense in the context of those patterns. We learn about combinations that make sense; this is useful because there are "far more of these nonsensical combinations than of the combinations which make sense" (Alexander, 1979, p. 207).

Connections to other patterns also tell us where to proceed if in a resulting context not all the forces are resolved: "Each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained" (Alexander, 1979, p. 312). Connections between patterns provide orientation in the pattern vocabulary. They make explicit that a "pattern uses another pattern, a pattern refines another pattern, or a pattern conflicts with another pattern" (Noble, 1998).

2.3.5.6 Processing a language

By knowing the vocabulary and its right application we can construct meaningful sentences, by knowing the right patterns and their appropriate combinations we can create meaningful designs (e.g. buildings and places, software architectures, educational scenarios). In other words, pattern languages support us in generating whole designs: "We usually use (or imply) the term 'generative' when speaking of pattern languages. Just as the English language can generate all possible papers in conference proceedings, so a pattern language can generate all sentences in a given domain" (Coplien, 1998d).

As we follow a sequence of pattern applications step by step we are passing through "The Gate" (Alexander, 1979). In this process we build the solution by piecemeal growth: "by using pattern sequences, a specific solution is developed stepwise through many creative acts until it is complete and consistent in all its parts" (Buschmann, Henney & Schmidt, 2007, p.298).

The structure of a language guides a designer to the next steps, and always lets him know where he currently is in the design process, providing the options for alternative design decisions. Each decision will influence the patterns that follow. A language "can guide you through the process of creating an end product of some kind. Each pattern resolves the forces in the current context. As it does so, it changes the context and new forces arise. You will need other patterns to resolve these new forces, and of course, the new patterns will change the context, and so on" (Rising, 1998, p. 56). A single pattern solves one design problem; a pattern language is capable of creating a complete structure.

2.3.5.7 Misunderstanding explicit structure for language

The idea of a pattern language is that patterns are not used in isolation but work together to form a larger whole. The relations between the patterns form a topological network, and the documentation of a pattern language usually explicates relations between the patterns: "Once we have understood how to discover individual patterns which are alive, we may then make a language for ourselves for any building task we face. The structure of the language is created by the network of connections among individual patterns" (Alexander, 1979, p. XII).

However, such explication bears a danger: people may interpret the relations as exhaustive. But this is, in my view, impossible. Of course, in natural language not all combinations of words make sense. Nevertheless it would be impossible to explicitly list all possible combinations for a vocabulary such as "tree". Rather, the power of human language is that we can create new and unpredictable meanings by the combinations of words. I think that we have to assume that for patterns as well. The power of a good pattern comes from its openness for combination with other patterns – without exhaustively naming all valid combinations. Therefore, I disagree with interpretations that a pattern language is defined wholly by explicit cross references between patterns. Such cross references are suggestions. They can point out a specific path to an end product, taking into account the new context after each design step.

The power of patterns of a pattern catalogue or a system of patterns is that they can be used to build many different end products. A pattern in Alexander's pattern language tells you what is built – for example a BUS

STOP. A pattern in *Design Patterns* only solves some aspects of a design problem. These patterns have to be used in combination with other patterns to create a whole solution.

We have to note that the combination of patterns is unlike plugging together Lego bricks (Quillien, 2008). Two patterns applied at the same time merge to a larger whole; their fields of forces interfere and the effects of one pattern feed back to the other. A hierarchical system of patterns is not automatically a language - in particular if the patterns of lower levels are just building bricks.

But not all of the patterns must be taken from a general pattern language. Actually, Alexander constantly refers to the specific pattern languages that he and his team created for particular projects. For a particular project it is indeed possible to describe the single patterns (the words) and all of their meaningful relationships (the grammar and semantic). For general patterns languages, however, we can only provide a vocabulary and, as in a system of patterns, some of the typical or common relations. There might be some general rules such as patterns of smaller granularity (design patterns) are used in the context of patterns of larger granularity (architectural patterns) and not vice versa. Courses (such as LECTURE, SEMINAR, or PROJECT WORK) are used in the context of educational organizations (a SCHOOL, or a UNIVERSITY) and not vice versa. But it is impossible to list all potential combinations of sub-patterns that are meaningful in the context of a SEMINAR. Nevertheless a context description may refer to typical combinations as examples.

If the "Context" and "Resulting Context" descriptions do not name other patterns they should describe the situation before and after applying the pattern in general terms. If the context of one pattern fits to the Resulting Context of another pattern they can be combined properly; this does *not* require explicit naming of patterns. A vivid pattern language to create a product or process is made by meaningful combinations of particularizations of general patterns and newly defined patterns. Noble, Biddle & Tempero (2006) conclude: "An encyclopaedia can be very similar to a pattern language in places. Where one compound pattern uses another pattern […] a structure like a pattern language is created in a localised part of the pattern collection, as and when it makes sense. Unlike a pattern language, this structure does not have to encompass the whole encyclopaedia, so a pattern author is not required to provide an initial pattern describing a single large-scale problem, to ensure all patterns are subparts of the initial pattern, or to omit patterns that do not fit."

The patterns in Design Patterns are not a pattern language (Gamma, Helm, Johnson & Vlissides, 1995, p. 366) but they are probably the most frequently used pattern descriptions so far. They provide import value without a complete order: "But even the relatively unorganized sets of patterns in the design pattern catalog are extremely valuable. Even when patterns are not organized into a language, they still can be used together synergistically" (Johnson, 1998a).

2.4 The quality of patterns[5]

Once we understand the descriptive components of patterns and the relations in a pattern language it is time to ask: what makes a pattern a good pattern? Which are the qualities of a pattern that we are looking for? Certainly we want that the pattern works and solves a problem to a satisfying degree. But a pattern should also be elegant, work with other patterns to form a solution on a larger scale, and be flexible to be used in various situations.

The next sections discuss the qualities of patterns that are whole and meaningful. They will draw from Alexander's views on wholeness as well as quality standards expressed by the pattern community, such as encapsulation, generativity, equilibrium of forces, abstraction, openness and composability (Lea, 1994). The path metaphor is still used to illustrate the concepts. However, the discussion will be more theoretical and focus on examples from the domains of software design and education.

In his later work *The Nature of Order* (Alexander, 2002) focuses on the wholeness of structures. He equals wholeness with the degree of life a thing has. Only things that are whole and alive possess the "Quality Without A Name" (Alexander, 1979) and resonate with a beholder. Since wholeness is the ultimate goal, we will start with a discussion of wholeness and then see how each of the other concepts will contribute to make or keep a pattern whole.

Each pattern is an abstraction of particular designs. Therefore, we will consider which abstractions sustain wholeness. An abstraction simplifies the matter and makes it possible to apply a solution to new situations. Abstraction also allows the composition of solutions by choice of appropriate alternatives. By not specifying every detail, an abstraction supports the variation and adaption to the specific needs of a situation. We shall

[5] An earlier version of chapter 2.4 has been published as "The structure of patterns – Part II" (Kohls, 2011).

therefore discuss which abstractions preserve the essential structure of a design without loosing wholeness, i.e. the pattern's gestalt. We are also in need of understandable and flexible ways to particularize an abstract form to a specific design. This will be covered in the sections composabilty and openness. Composability will show how an abstract form can be specified by composition of more specific forms. Openness will discuss to which degree an abstract determines the composition or is open to variations.

In order to maintain wholeness, each of the combined and adapted parts has to fit to all of the other parts. Fitness also means that the parts have to properly interact. A single misfit could disturb the balance or equilibrium of the whole design. We will discuss fitness and equilibration in the succeeding section. Making each design decisions fit to all the other design variables leads to an explosion of complexity that needs to be handled. Encapsulation is a way to make parts nearly independent, thus one can focus on each design problem at a time without loosing the general picture.

Encapsulation could be seen as the counterpart to composition. While composition combines sub-wholes to a larger whole, encapsulation divides a larger whole into smaller self-contained sub-wholes. On a more abstract level we do not have to focus on the particular configuration of a sub-whole. On a more specific level we have the choice between alternative configurations that serve the same goal. We can choose the most appropriate alternative for the particular situation. Depending on the openness of the pattern, a multiplicity of design choices is available to unfold a particular design. This process of unfolding is tackled in sections about generativity and evolution. Finally, we will discuss that patterns have to be meaningful to humans.

2.4.1 Wholeness

2.4.1.1 Alexander's views on wholeness

Alexander's quest for wholeness can be traced back to his early work *Notes on the synthesis of form*: "[the design process] concentrates on structure, the process is able to make a coherent and therefore new whole out of incoherent pieces" (Alexander, 1964, p.110). It is discussed widely in *The Timeless Way of Building*, where the interplays of patterns are means to form whole structures: "the language lives, or not, as a totality to the degree these patterns form a whole" (Alexander, 1979, p. XII). The previous quote hints to Alexander's understanding of wholeness as a degree of life, which is elaborated in *The Nature or Order*: „In order to understand life as a phenomenon, it is necessary to define something which I call ‚the wholeness'" (Alexander, 2002a, p. 80). There are also references to gestalt theory, " […] the fact that the level of wholeness of different centers is objectively given, and may in principle be determined, was described by the gestalt psychologists Max Wertheimer, Wolfgang Köhler and Kurt Koffka, who formulated the laws of 'praegnanz' as the determining features of a whole, which gives its strength" (Alexander, 2002a, p.107).

2.4.1.2 Practical implications of wholeness

What does wholeness mean in practice? It certainly means that any solution has to work completely and as a whole. Every programmer is well aware that errors or design failures in the small can cause the whole system to break. Each part of the system has to be designed in a way that it serves the need of the whole. Wholeness means that we are not considering software modules or objects in isolation. An encapsulated component can help to reduce the complexity of designs; hence, libraries are means to re-use algorithms and code. However, if a library cannot be adapted to the specific needs of a situation it cannot serve the whole appropriately: "it is impossible to form anything which has the character of nature by adding preformed parts" (Alexander, 1979, p. 368). The parts of a software design have to fit together across the whole architecture. Patterns capture structures and relationships that are not limited to isolated parts: "These deep components of architecture and design are larger than any architectural building blocks such as procedures or objects" (Coplien, 1996). Wholeness means a balance between the inner forces and the environment; all parts of a system have to be in balance with all the other parts at the same time (Gabriel, 1996).

2.4.1.3 The interplay of parts

But wholeness also means that a system has a new quality that cannot be reduced to its singular parts. This is closely related to gestalt theory: "A system cannot be taken for the sum of its parts; indeed, formal analysis of artificially isolated segments would destroy the very object of interest. It is necessary to neglect the parts of the gestalt and attend to the core of its complexity, its organization. The psychological concept of gestalt is only one way of expressing the principle of nonsummativity; in other fields there is great interest in the emergent quality that arises out of interrelation of two or more elements" (Watzlawick, Bavelas & Jackson, 1967, p. 125).

A whole form emerges out of the dynamic interplay of smaller whole forms and serves itself to a larger whole. For example, each part of a software system has to be designed in a way that it serves the need of the whole in order to give it a gestalt: "the concept of 'need of the whole' refers to the grand designs or architecture of the piece of software under development, and 'needs of the parts' refers to the inevitable changes the various parts of the software undergo. It's difficult to change the grand design of software: You cannot expect to evolve a window system into a spreadsheet. Although the primary need of the whole is to remain true to its essence, the parts often must change. For instance, one sort of window system could evolve into another" (Gabriel, 1996, p.13).

2.4.1.4 Examples

In education, a blended learning scenario consists of whole parts such as INITIAL MEETING, ONLINE SEMINARS, DISCUSSION FORUMS, SHARED GROUP SPACE and many others. While each of these parts can be considered as a self-contained form, the whole scenario is more than the sum of these parts. Something new emerges and one gets more out than was put in. However, what is very often overseen is that the parts too depend on the whole, i.e. on the environment. In a holistic view each part reflects the whole to certain degrees. For example, the blended learning scenario does not only have an INITIAL MEETING but the INITIAL MEETING itself fully depends on the scenario. For example, if there are ONLINE SEMINARS and DISCUSSION FORUMS planned then these parts will influence what happens in the INITIAL MEETING (e.g. make a demonstration how to use the supporting online tools). Likewise, the INITIAL MEETING will have an impact on how these online tools will be used (e.g. a poor introduction or motivation may lead to very rare use); what happens in the SHARED GROUP SPACE depends on the other parts as well, such as the INITIAL MEETING or announcements in a DISCUSSION FORUM. It shows that the whole blended scenario emerges not from the single components but from the interplay of connected parts. A form without context is not a whole form – what is the meaning or value of an EDUCATIONAL BLOG if it is not used for a purpose in a specific situation? Hence, each part gets its character to some extent from the whole.

Accordingly we cannot consider any part without its context. That means the parts are giving the whole its character but at the same time the whole context influences the meaning of the sub-wholes: "Wholeness is seen as primary while the parts are secondary in the sense that what they are and what they do can be understood only in the light of the whole. I could summarize this in the principle: The wholeness of the whole and the parts" (Bohm & Factor, 1985, p. 22).

The whole does not come after the parts but is rather primary in that it organizes the parts, making them work together, and effectively influences what the parts are. It is a design as a whole that orchestrates how the single parts are unfolded. However, the parts are not determined by the whole – this would make the whole a super-part: "…a part is a part only inasmuch as it serves to let the whole come forth, which is to let meaning emerge. A part is only a part according to the emergence of the whole which it serves; otherwise it is mere noise. At the same time, the whole does not dominate, for the whole cannot emerge without the parts. The hazard of emergence is such that the whole depends on the parts to be able to come forth, and the parts depend on the coming forth of the whole to be significant instead of superficial" (Bortoft, 1996, p. 11). We experience wholeness if we follow a story in a novel. The plot unfolds chapter for chapter, paragraph for paragraph, sentence for sentence, word for word. The parts make the story and the story gives meaning to each of the parts. A simple sentence such as "The door was locked" has its own meaning; however, in the context of a larger story its meaning can alter. A locked door has a deeper meaning in a crime story where a victim tries to escape. The same sentence can have a different meaning in a love story: "She wanted to tell him her feelings and caught up with the train at the local station. The door was locked." The context does not only change the meaning of the sentence. A single sentence that reveals an important fact or event can also change the meaning of the whole story. The story directs the development of the events, scenes and characters; at the same time the story is made exactly out of these interrelated parts.

2.4.1.5 Situatedness

The context not only influences how the "inner" parts have to be orchestrated but actually changes what these parts are in a field-like effect. Consider figure 11 in which we use the very same circles in two different contexts. In context (a) the circles are eyes, in context (b) they are part of loudspeakers. Hence, what the very same form is depends on the context. Yet the form itself contributes to the context – without the circles we would neither see a face nor the loudspeakers as (c) or (d) show. Schümmer (2005a) draws the analogy to Heidegger's phenomenology: “Situatedness (In-der-Welt-sein) describes a situation where the individual cannot understand his being without taking the current context, the situation, into account.” (Schümmer, 2005a, p. 10).

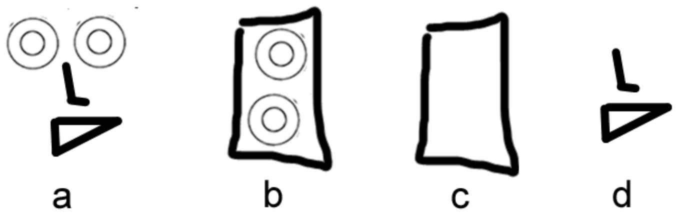


Figure 11. The meaning of the circles depends on their contexts.

In analogy, the same structural form of a DISCUSSION FORUM (e.g. the same software tool) takes a different gestalt (a different whole) when used in an educational or a corporate context. The blended learning scenario emits to the DISCUSSION FORUM, and the DISCUSSION FORUM contributes to the scenario.

2.4.1.6 Nesting of wholes

At the same time each design is also part of an even larger whole, e.g. the blended learning scenario is embedded in a curriculum or a post-graduate study. As such it depends on the curriculum while actively taking part in what that curriculum is. Likewise, the curriculum is embedded in an even larger context. For example, the university’s observation of which skills are demanded by a society will suggest which curriculums should be offered. This nesting of wholes and parts is core to a holistic view and to design: “This recursive whole-part nature can be seen as a universal theme in the realm of design. We view systems recursively in this way because the whole is more than the sum of its parts, otherwise we would not need to support multiple perspectives” (Buschmann, Henney & Schmidt, 2007, p.180). The nesting of wholes implies that the designer has to mind the fitness “at several boundaries within the ensemble at once” (Alexander, 1964, p.18) and that “there is a perfect balance between the needs of the parts and the needs of the whole” (Alexander, 1975, p.14).

2.4.1.7 Whole paths

Let us interpret wholeness for a path. At no point of the path should there be an obstacle that cannot be managed. The path has to fit to its environment at all times. A single misfit – such as a blocked passage or a deep river without a bridge – will invalidate the whole path. The path also has to connect properly to other paths in order to be open for the composition of larger paths.

A path can also be divided into sub-paths or segments. However, not any partition of the whole path into sub-paths is meaningful. For example, in figure 12 if we take an arbitrary sub-path between point A and B, then this path is hardly self-contained.

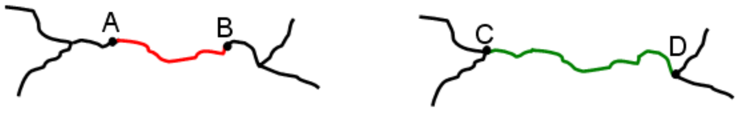


Figure 12. Wholeness of a path

There is only one way getting to A and only one way to continue from B. The sub-path cannot stand on its own. The path is not a whole sub-segment. However, at points C and D the path connects to multiple other paths. The section between C and D is more whole because it can stand on its own – it is a whole solution. It makes sense in multiple situations, i.e. coming from different paths and following on different paths afterwards. To be whole it has to properly connect to environmental paths at point C and D. But its inner course no longer depends on C and D. It can be shaped nearly independent according to the local surroundings.

2.4.2 Abstraction

Abstraction means to reduce a structure to the elements that matter by leaving out details that are considered irrelevant in the given context. In the discussion of wholeness we have learned that everything matters. So, how can we leave out details if everything matters?

2.4.2.1 Invariant structure

Patterns are abstractions of instances that share similar structures: "Patterns represent abstractions of empirical experience and everyday knowledge. They are general within the stated context, although not necessarily universal" (Lea, 1994). The context has an important impact on the level of abstraction. A general context implies that the solution is also stated in more general terms. The reason is that there will be many differences in the particular contexts and therefore the solution must be flexible to adapt to the particular forces found in the situation.

Since the context implies which features of a solution are relevant it also implies what can be omitted. An abstraction is a mapping of some part of the world that omits irrelevant features. However, if we choose an abstraction that loses essential information nothing is gained. Abstraction and encapsulation should be used only "in moderation" (Gabriel, 1996, p.19). Patterns are mid-level abstractions and not blueprints or exact step-by-step recipes: "much of the power of patterns stems from the fact that they do not prescribe a particular implementation" (Buschmann, Henney & Schmidt, 2007, p.76). They are rather sketches or loose diagrams that illustrate the structural quality of the pattern, i.e. the fundamental relations of the elements of a pattern. A pattern does not specify every detail but its gestalt needs to sustain: "It means, of course, that I want to make a simple picture of it, which lets me grasp it as a whole. And it means, too, that as far as possible, I want to paint this simple picture out of as few elements as possible. The fewer elements there are, the richer the relationships between them, and the more of the picture lies in the 'structure' of these relationships" (Alexander, 1979, p. 81).

2.4.2.2 Different ways of abstraction

There are many ways of abstraction but let us consider three basic kinds:

- abstraction by isolation, i.e. omitting features from a structure that are not relevant
- abstraction by generalization, i.e. omitting irrelevant variations of structural features
- abstraction to emergent qualities, i.e. omitting the structure of a phenomena in favour for its emergent effect

The difference between isolation and generalization is discussed by Wundt (1907). Abstraction to emergent qualities is discussed in the context of both information and complexity theory (for example Holland, 1998; Mitchel, 2009; Gleick, 2011) but the idea of complex abstraction has already been discussed by St. Thomas (Bobik & Sayre,1963). A more philosophical discussion about these types of abstraction can be found in section 3.2.7.

The first two abstractions can be instructive as long as we omit no information that is relevant for the given context. But we must be careful to not loose the gestalt. The third kind of abstraction is trickier because it abstracts from the actual structure to its effects. Since two different structures can have the same effect (e.g. two different paths lead to the same goal), the abstraction does not capture the essence of the underlying structure but only its quality. The differences of structure may not matter on a higher level of abstraction (macro-level) but they may matter on a lower level of abstraction (micro-level).

To illustrate the various abstraction forms, let us consider the following shapes of visual patterns in figure 13.

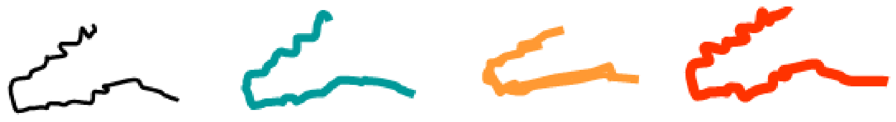


Figure 13. Shapes with different features.

Each shape differs in some features, such as colour, thickness, or texture. However, if we are only interested in the geometric points of the shape, we could isolate the form without considering these other features.

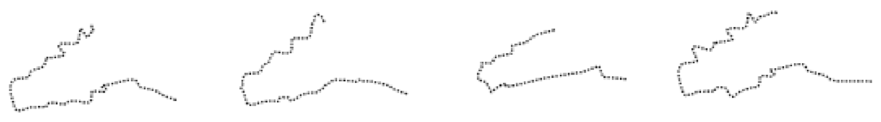
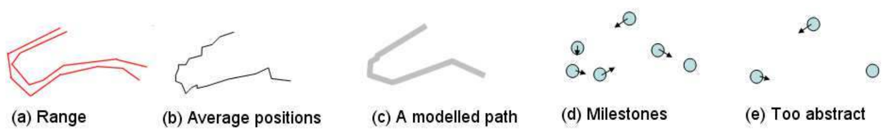


Figure 14. Abstraction by isolation: Shapes abstracted to their geometric points.

Though each of the shapes is still different, they seem to be very similar in their structure. The shapes seem to share the same general form or gestalt. Thus, we can abstract by generalizing the form. This can be done in different ways, using different approaches to generalization, see figure 15.



Figures 15. Abstraction by generalization: Different Generalizations.

Generalization (a) shows the spatial range in which all example shapes occur.
Generalization (b) shows a shape that is based on the average positions of all the other shapes.
Generalization (c) shows a modelled shape that resembles the gestalt and shows its simplest manifestation.
Generalization (d) shows only the points in which the direction changes.
Generalization (e) shows an abstraction that no longer accounts for the essential features of the shape.

Generalizations (a)-(d) are valid to preserve the general gestalt of the shapes. (a) spans the space in which all the manifestations of the shapes take place; this is similar to patterns that describe a space of potential designs rather than describing a specific instance. (b) and (c) are both mappings to an abstract example of the shapes. Though they are not among the original examples of the shapes they could potentially be – for they are both within the range of possible manifestations. The modelled example seems to be a more illustrative choice because it is simpler but not less expressive. Modelled examples can serve very well in pattern descriptions to illustrate and exemplify the essence of the pattern. (d) shows the essential milestones of the modelled shape. It is reduced to the directions required to follow the path. Such milestones would be essential to pattern descriptions that try to instruct how the gestalt of a pattern can be generated. Generalization (e) lacks some of the essential milestones. As a consequence, the resulting gestalt is different.

A more detailed zoom into the structure of the original shapes (figure 16) reveals that what on a higher level of abstraction seems to be the same could indeed have very different forms on a lower level of abstraction.

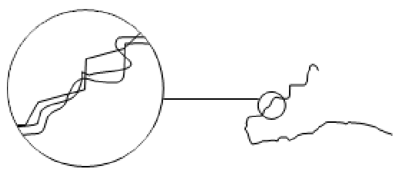


Figure 16: Abstraction to emergent qualities: Different wholes are considered to have the same quality on a higher level of abstraction.

The two sub-shapes are very different but on the higher level of abstraction this difference does not matter. For the gestalt on a higher level of abstractions depends not on the detailed structure.

2.4.2.3 Examples

The illustrated shapes could be mappings of paths in our world. Indeed the shapes can be found in our hillside landscape (see figure 7). Let us consider the effects of abstraction when we provide a description of path directions. On a high level description of a path there could be an instruction “Cross the bridge”. For the general description of the path it does not matter how to cross the bridge: whether one crosses the bridge on the left side, in the middle or the right side is not specified.

However, in the actual crossing of the bridge, the details do matter: if there is an obstacle at the right hand side, one can only cross at the left hand side; if it is stormy or icy one should better move slow and do not rush. The local adaption requires that the hiker knows how to achieve the abstract goal "Cross the bridge".

A more abstract description of the same path could include the instruction "Cross the river". This instruction omits the information how and where the river should be crossed: via the bridge, using a boat, swimming or jumping from stone to stone. Each option has the same abstract quality in respect to the goal of "crossing the river". Yet crossing a bridge or swimming over the river are certainly different forms. Therefore, the instruction "Cross the river" does not tell the whole story. To make it whole one has to know the available options and make one appropriate decision. If none of the options is known to the recipient of that information, the instruction "Cross the river" remains too abstract. But if the options are known, the more abstract notion is more open as it provides choices to compose the actual whole. In education the descriptions of methods and models are often either too abstract or too specific (Baumgartner, 2011). For example, if we describe a scenario and refer to GROUP WORK we are describing a social form on a high level of abstraction for there are many different forms of group work. For an expert this abstract notion is more open because s/he can choose between the different forms. A novice, however, is better of with a more detailed description how to organize group work. While a detailed description is more instructive it narrows the flexibility and does not provide alternatives. Pattern languages or a network of patterns can serve both needs. On a high level description of a scenario one can refer to GROUP WORK as another pattern. Experts know already what that means while novices can consult the pattern which elaborates the structure of group work in more detail and points to further sub-patterns (e.g. BUILD RANDOM GROUPS, GROUP BY SKILLS, PRESENT RESULTS). Thus, abstraction allows the composition of solutions based on prior knowledge and consultation of alternative patterns.

2.4.3 Composability

2.4.3.1 Composition of patterns

Patterns are hierarchically related to compose larger wholes. The relations between patterns and their suitable combinations are made explicit in pattern languages. Solutions are not build from patterns in isolation but composed: "Most patterns are both upwardly and downwardly composible, minimizing interaction with other patterns, making clear when two related patterns must share a third, and admitting maximal variation in sub-patterns. Pattern entries are arranged conceptually as a *language* that expresses this layering" (Lea, 1994).

A curriculum does not only contain one course type but a mixture of different formats that can be combined with each other. At a lower level, a SEMINAR could use a combination of different methods to deliver concepts. This network of related patterns forms a pattern language. The singular patterns are the vocabulary, the relations and the proper forms of combinations are its grammar. As with natural language not all combinations make sense. That is why many pattern descriptions explicate "related patterns".

2.4.3.2 Composition of paths

A path can consist of smaller segments of sub-paths as figure 17 shows. If X and Y both are means to the same goal on the next higher level of abstraction, we can say that they are properly encapsulated in a way that they both have the same effect. Path X and Y can be qualitatively different on a micro-level but qualitatively equal on a macro-level.

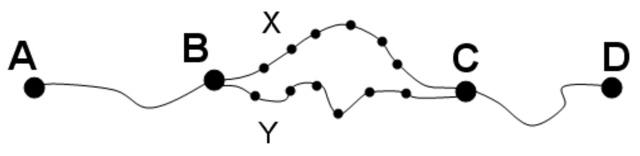


Figure 17. Example: The paths X and Y have different forms; the succeeding points follow a different structure. On a higher level, however, X and Y serve the same goal. To follow the path A-B-C-D, it does not matter whether X or Y is chosen.

The abstract "Drive from Hamburg to Munich" can be expressed by "Drive to Hannover, Göttingen, Kassel, Fulda, Würzburg, Nürnberg, Ingolstadt, and then to Munich". Each single milestone could then be further specified by a more particular route, e.g. "To get to Hannover, drive to Seevetal, Fallingbostel, and then to Hannover." A benefit of the abstract representation is that we can replace each of the abstractions by different specializations to compose the whole. If the specializations are properly encapsulated, then we can independently choose an option. For example, the abstract "Drive to Hannover" can also be replaced by the alternative route "Drive to Lüneburg, Uelzen, Celle, and then to Hannover".

2.4.4 Openness and variability

To have a choice between alternative sub-wholes that integrate into the larger whole is one sign of openness. This openness is important in order to take the specifics of an actual situation into account. Openness refers to the agility and adaptability of patterns as well as their compatibility with other patterns.

Alexander points out that nature is never modular; there are "similar units (waves, raindrops, blades of grass) – but though the units of one kind are all alike in their broad structure, no two are ever alike in detail" (Alexander, 1979, p. 144). The general patterns need to be differentiated according to the specifics of the situation and fitted to their individual circumstances. Modular parts or premade entities can hardly account for the specifics of a situation.

2.4.4.1 Combination of compatible patterns

Patterns are nested and "may be extended down to arbitrarily fine levels of detail. Like fractals, patterns have no top or bottom -- at the lowest levels of any design effort, some are merely opaque and/or fluid (e.g., plaster, concrete)" (Lea, 1994). In order to adapt to the specifics of a situation, a pattern is more open if it allows a variety of sub-patterns. A pattern that does not narrow the options how to proceed is more open. A resulting context that can be addressed by more than one sub-pattern is more open. Likewise, a context that is broader (e.g., that includes more previous or next steps) makes the pattern more open for the combination with other patterns.

The selection of a path that is not open means that we have to "stick" to a particular configuration. An example for such a configuration is a railway track. Once a train is on track it has to stay there until the next switch. If a tree should fall on the railroad, the train is stuck until the tree is taken away. A regular hiking path is more open: if a tree is on our way we can walk around or climb over it. The openness of a path may vary at different points. A narrow section close to a sheer offers less space for variation. A meadow allows running criss-cross. A high number of connecting paths is another indicator of openness.

2.4.4.2 Densely overlapping patterns

Another degree of openness is achieved if a pattern does not only allow the combination of patterns in an additive way (i.e., combination of parts) but in a multiplicative way (i.e., overlapping of parts). Gabriel (1996) compares the overlapping of patterns with compression of words that have many meanings. In software design, a class can be part of multiple patterns at the same time because "it is not simply that one patterns is applied after another, it is that their roles are compatible and can be integrated" (Buschmann, Henney & Schmidt, 2007, p.197). In a teaching scenario, a specific resource or activity can be an element of multiple patterns (e.g. a VISUAL ILLUSTRATION can at the same time serve the patterns MOTIVATION, INTRODUCTION and providing the BIG PICTURE). Alexander highlights the importance of a dense use of patterns: "It is possible to make buildings by stringing together patterns, in a rather loose way. A building made like this, is an assembly of patterns. It is not dense. It is not profound. But it is also possible to put patterns together in such a way that many patterns overlap in the same physical space: the building is very dense; it has many meanings captured in small space; and through this density, it becomes profound" (Alexander et al., 1977, p. XLI). The Gang of Four cite the previous statement at the end of their book to highlight that "the best designs will use many design patterns that dovetail and intertwine to produce a greater whole" (Gamma, Helm, Johnson & Vlissides, 1995, p.358).

2.4.4.3 Modifying patterns

The merge of patterns into a new whole is rather like cooking than building something by connecting Lego bricks: "the metaphor of cooking has more to offer us than the metaphor of construction: ingredients blend and affect one another in a way that lacks convenient parallels in the world of steel, glass, bricks, and wood" (Buschmann, Henney & Schmidt, 2007, p.185). This means that the very character of one pattern can be altered by other patterns. Just as an apple tastes different in diverse meals, an OBSERVER may have different emergent properties depending on other patterns at work.

Kolfschoten & Santanen (2007) introduce the concept of MODIFIERS, patterns that cannot exist on their own but modify other patterns. This is like the property red that can modify the colour of an object but it always needs other objects to manifest. A MODIFIER can reduce the number or patterns in a pattern language (or system of patterns) because invariants between patterns are extracted as patterns on their own.

2.4.4.4 Different types of openness

Thus, we have seen three kinds or levels of openness:

- Combination of patterns
- Overlapping patterns
- Modifying patterns

The combination of patterns means that a pattern can be part of many other patterns and it could be made of many other patterns. For example, a TABLE could be used in many room arrangements and it could be made of different parts. But its parts (e.g. its legs) cannot overlap with other objects in the room. Overlapping means that the same elements are part of multiple patterns at the same time, for example one TABLE can be an element of WINDOW PLACE and CHAIRS AROUND TABLE at the same time. A MODIFIER such as *Redness* can change the character of other patterns, (i.e. a RED TABLE) but it can only be exemplified in combination with other patterns.

2.4.5 Equilibration and fitness

The combination of patterns and design decisions requires that each of the sub-wholes has to be in harmony with all of the other sub-wholes. Each sub-whole has to fit to its current environment. That includes other patterns of the same level of abstraction as well as patterns on higher and lower levels of abstractions. An equilibrated state of the whole is a configuration of interrelated forms that minimizes the conflict among forces and constraints of the environment. That is why a pattern description should always discuss the forces to illuminate how the solution of a pattern resolves them to an equilibrated state. In Alexander's view, a pattern can only be fully alive if all the forces in a situation are dealt with: "If the adaptation to the forces is not perfectly exact, there can be no comfort, and no freedom, because the small forces which have been left out will always work to make the system fail" (Alexander, 1979, p. 34). The goal of equilibration should justify each of the design steps.

2.4.5.1 Equilibrium of forces in pattern languages

All potential implementations of the same pattern contain an invariant that minimizes the conflict among forces and constraints (Lea, 1994). Alexander claims for his pattern language that the conflicts fully resolve by the interplay and combination of patterns. In the software pattern community it is appreciated that a single pattern never fully resolves all forces. Unresolved forces are usually discussed in the consequences section.

That a single pattern does not resolve all forces may not be problematic because patterns always require other patterns to create a whole solution. Patterns influence each other, and the "individual configuration of any one pattern requires other patterns to keep itself alive" (Alexander, 1979, p. 131). The equilibrium is achieved not by applying a single pattern but multiple patterns together. Stable configurations of smaller patterns create a pattern on a higher level that itself achieves a state of equilibrium.

The unresolved forces of a design pattern explicate the demand for further resolution. The patterns in Alexander's pattern language miss the explication of unresolved forces because each pattern links to other patterns in order to stabilize it. In other words: a single pattern of Alexander's language only becomes balanced if it is applied in conjunction with other patterns of lower, higher and the same level.

A pattern language combines pattern configurations that are themselves stable, alive, balanced and whole: "A pattern language must also balance forces related globally to its subject as a whole, rather than just to individual problems that arise when creating specific designs or implementation for this subject" (Buschmann, Henney & Schmidt, 2007, p.275).

The global balance of forces suggests that applying a single pattern of a pattern language without applying the other patterns will not lead to equilibrium: "The quality without a name occurs, not when an isolated pattern occurs, but when an entire system of patterns, independent, at many levels, is all stable and alive" (Alexander, 1979, p. 131). If we just grab out one pattern we have not resolved all forces – for that we need to apply other patterns from the language or create our own solutions that address the remaining forces. In this case we create our own pattern language for a particular solution. It is not necessarily a pattern language with recurrence. It may be a particular pattern language in which the single solution structures (particularized general patterns and new particular patterns) work in full harmony. Grabbing out just one pattern from a (general) pattern language is no better than taking a pattern from a pattern catalogue. In fact, if we take a pattern from a pattern catalogue we have to find our own (particular) pattern language to support it. The consequences section of pattern descriptions provides a lot of suggestions how to do this by telling us what is still needed. This makes the application of patterns of a catalogue much more flexible (Noble, Biddle & Tempero, 2006).

2.4.5.2 Fitness between form and context

Another way of saying that the forces are in equilibrium is to say that the forces of the solution fit to the forces of the context. We can speak of a solution – a resolution of forces – only if we consider both the form of context and the form of the solution. The idea of fitness between two entities is most clearly illuminated in Alexander's early work *Notes on the Synthesis of form* (Alexander, 1964). To balance the forces, the context and solution form are undividable in consideration: "It is based on the idea that every design problem begins with an effort to achieve fitness between two entities; the form in question and its context. The form is the solution to the problem; the context defines the problem. In other words, when we speak of design, the real object of discussion is not the form alone, but the ensemble comprising the form and its context. Good fit is a desired property of this ensemble which relates to some particular division of the ensemble into form and context" (Alexander, 1964, p.15).

Each step in a design process is a variable decision that can fit or misfit. Along a path each step is a decision into a direction. Each step has to fit to its current environment. A step into a canyon or a deep river is a misfit; the chosen path does not work as a whole. At the same time each step depends on the previous steps. A misfit indicates that not all of the forces are in balance.

2.4.5.3 Complexity of design problems

The interrelation of variables makes the right configuration a complex design problem. Alexander uses the example of a matrix of light bulbs that are interrelated and their state represents a misfit (on) or fit (off): "The lights are so constructed that any light which is on has a 50-50 chance of going off in the next second. […] Connections between lights are constructed so that any light which is off has a 50-50 chance of going on again in the next second, provided at least one of the lights it is connected to is on" (Alexander, 1964, p. 39). The next state of a bulb depends on its own current state and on its environment as well: "Sooner or later the system of lights will always reach this equilibrium. The only question that remains is, how long will it take for this to happen? It is not hard to see that apart from chance this depends only on the **pattern** of interconnections between the lights" (Alexander, 1964, p. 40, highlighting added). In this passage we see a first occurrence of the term "pattern" in it its later meaning. If there are no connections at all (i.e., lights can change independently) we only need two steps (2^1) on average to have all lights turned off because once a light is off it remains off. On the other hand if all lights are interconnected, then a single light that is on can turn on all other lights in the next step. The only chance to get a stable system is that all lights turn off by chance at the same moment – this will happen on average after 2^n steps.

Both cases are unrealistic; any system does have interrelations of variables but if all variables would relate to each other we could never solve any problem in feasible time. However, we can "discern in the **pattern of interconnections** some 10 principal subsystems, each containing 10 lights. The lights within each subsystem are so strongly connected to one another that again all 10 must go off simultaneously before they will stay off; yet at the same time the subsystems themselves are independent of one another as **wholes**, so that the lights in one subsystem can be switched off without being reactivated by others flashing in other subsystems" (Alexander, 1964, p. 41, highlights added). Dividing a system into subsystems or sub-wholes is what is achieved by encapsulation.

2.4.6 Encapsulation

Encapsulation is a means to solve the problem of complexity (Alexander, 1964). A complex system needs to be decomposed properly into sub-problems that can be solved one at a time and that are well understood without loosing the big picture of the general problem frame.

2.4.6.1 Decomposition into self-contained wholes

The subsystems are considered as self-contained wholes; the structure of such a whole is its pattern of interconnections. A pattern captures a part of a system that has a large number of inner connections and only a small number of connections to its environment. Such configurations are nearly independent (Simon, 1962).

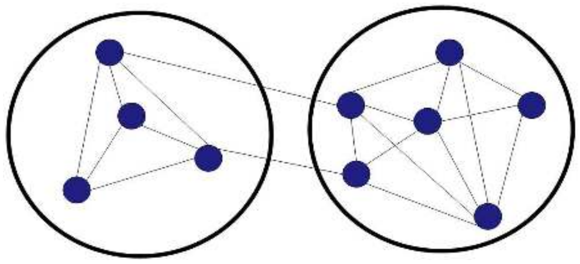


Figure 18. Within the partitions there are many interrelations among the elements, making each part self-contained to some extend. The environment, however, governs the form of a whole part at the boundaries. Figure based on Alexander (1964, p. 65).

The question is which parts of the whole are themselves whole parts? Alexander (1964) illustrates the encapsulation and reduction of complexity of a (design) problem by concentrating on those design variables that have a large number of interconnections but relatively few connections to their environment. The high frequency of connections is later termed by Alexander as centres (Alexander, 2002a), making it clearer that they are not entirely separated from the environment.

Every change within such a part will have relatively low impact on the surrounding context making it possible to explore different solution forms independently to some extent. Yet, the context does set the stage for the configuration of the form at least where there are interrelations. The “right part” is a part that fits to the whole in that it actually mirrors and entails the whole.

2.4.6.2 Examples

Encapsulation is a very important principle in software design. Large software systems can consist of several thousand classes. If each change would affect the whole architecture it could not be maintained. While wholeness suggests that the parts have to fit together and orchestrates the shape of a part, the inner structure of a part is nearly independent: “Patterns are independent, specific, and precisely formulated enough to make clear when they apply and whether they capture real problems and issues, and to ensure that each step of synthesis results in the construction of a complete, recognizable entity, where each part makes sense as an in-the-small whole” (Lea, 1994). All of the software design patterns are sub-wholes that have a general recurrent form that can be adapted to the specific needs of a context.

In pedagogy, teaching scenarios such as lectures, seminars, trainings, or assessments can be considered nearly independent. While their particularization depends on the whole context (curriculum, students, teachers) they have a self-contained form. In an e-learning scenario the interrelations of each of the parts of a SHARED GROUP SPACE are stronger (more centered) to other parts of the SHARED GROUP SPACE than they are to the outer context. Hence, any change of one of the parts will have stronger impact on the SHARED GROUP SPACE than it does have on the outer context, for example a blended learning scenario.

A path between two cottages is another example for an encapsulation. The shape of the path from cottage A to cottage B only depends on the environment between the locations of the two cottages. It does not matter to the shape of the sub-path how a hiker gets to cottage A and how he proceeds from cottage B (see figure 17). We say nearly independent because the sub-path does depend on previous and following paths to a certain extent. For example, if a group of hikers chose a path to cottage A that allowed them to bring a lot of supplies, they can choose a longer path from A to B. If the sub-path is exhausting this could have consequences for the selection for the next path followed from cottage B.

2.4.7 Evolution of patterns

The encapsulation of sub-wholes also means that each solution can evolve independently without deeply affecting its environment. For example, a door can change its design as long as it fits to the outer frame. The hierarchical decomposition and encapsulation leads to a faster evolution of each of the parts. Simon (1996, p.

196) shows that "complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not."

2.4.7.1 Best of breed

The term "fitness" suggests that there is a connection to the theory of evolution; good design would be considered as the survival of the "fittest" or "Best of Breed" (Buschmann, Henney & Schmidt, 2007). It has sometimes been suggested that the stable patterns in *A Pattern Language* are stable because they emerged over time and only the beautiful buildings have survived. Alexander often shows older buildings as good examples. One could think that old buildings have last because of their beauty while bad buildings have been cut down: "The last effect is that today we see those buildings and towns that have survived because they are pleasant – there is a natural selection. It would be odd to see towns and buildings in Europe that are old and just plain ugly; they would not have survived" (Gabriel, 1996, p. 41).

Alexander, however, denies an evolutionary view that focuses on progress based on random adoptions and natural selection. Rather he sees a necessity of forms (Alexander, 1964; Alexander, 2002). This is coherent with D'Arcy Thompson's view on evolution that assumes a natural (physical) necessity based on forces (Thompson, 1942). Yet Alexander himself points out that patterns are established cultural structures: "Patterns are structures that have evolved over ages of development to fulfil cultural needs. They are rules, driven by principles, that serve human and social needs" (Alexander, 1979).

2.4.7.2 Development, improvements and new facts

Patterns will also naturally evolve if we have a better understanding of their nature: "Patterns are living things that change as we learn more about the problem, the context, the solution. Some may completely outlive their usefulness as procedures and other technologies change" (Rising, 1998, p. 54). The patterns in the world evolve as well as pattern descriptions evolve. Pattern descriptions are always subject to revisions as we learn more about the patterns in nature and their environments – which evolve as well.

A path may improve over time or adapt to new environments. A known path is a working solution to reach a goal. But it is not necessarily the best solution. New paths can be discovered at any time and existing paths may be improved by experimenting with variations. For example, one can find a shortcut between two sections and over time people adapt to using the shortcut more frequently until the original path is not taken any longer and eventually "dies". Changes of the environment also impact the fitness of a path. New obstacles such as falling rocks can change the course of a path. A new bridge built over a canyon or river also changes the context and affords new ways.

2.4.7.3 Piecemeal growth and local adoptions

An evolution of forms, however, only considers the end product. Yet there is a difference between the phenotype (the physical appearance of a form) and genotype (the "program" that unfolds the form). The structure of the generating process is different to the structure of the generated thing. Alexander compares the process of unfolding to the "evolution of an embryo, in which the whole precedes its parts, and actually gives birth to them, by splitting" (Alexander, 1979, p.365). The development of cells and structure depends on the local surroundings as well as the organism as a whole; different contexts activate different genes to unfold the structure (Ball, 2009).

In this view, every solution unfolds step-by-step in a way of piecemeal growth and local adoptions. At any time in the process of design the current situation or context needs to be evaluated to progress. Each design step (or application of a pattern) changes the overall situation. The results cannot be fully foreseen and must be re-evaluated in order to select the next step. Thus, a solution form that fits to a particular context can only be produced indirectly (Gabriel, 1996, p. 50). Such pattern descriptions are called generative: "A generative pattern is a means of letting the problem resolve itself over time, just as a flower unfolds from its seed" (Coplien, 1996).

The term "memes" refers to the cultural evolution and replication of ideas in analogy to genes in the biological world. The term was originally introduced by Dawkins (1989). By variation and mutation the same "memes" can generate creative forms based on the same origin (Csikszentmihalyi, 1996). That patterns play a comparable role if they provide strong generative rules has been pointed out by Schümmer & Lukosch (2007).

2.4.8 Generativity

A design pattern does not only capture good solutions in order to understand them; it also captures the ways of generating the solutions (Coplien, 1998). A pattern is at the same time a thing and process, "a rule which describes what you have to do to **generate** the entity which it defines" (Alexander, 1979, p. 182). A generative pattern is understood to be a pattern that not only shows the positive invariants of working solutions but also teaches how to build them (Appleton, 2000).

2.4.8.1 Generativity of pattern languages

A pattern language is one way of taking the variation of the situation into account. Alexander sees pattern languages as "The Gate" through which the "Quality Without A Name" can be achieved (Alexander, 1979). By passing through the gate, i.e. the application of one pattern after another, we find ourselves constantly in new situations. The network of a pattern language guides a designer from situation to situation, tackling the concrete forces of the various levels: "The successive application of several patterns, each encapsulating its own problem and forces, unfolds a larger solution which emerges indirectly as a result of the smaller solutions. It is the generation of such emergent behaviour that appears to be what is meant by generativity" (Appleton, 2000).

The application of patterns in an orderly sequence in meaningful combinations has been acknowledged by the pattern community as pattern sequences (Harrison & Coplien, 2002). Alexander stresses, in particular in *The Nature of Order* (Alexander, 2002b), that there are sequences that are more fruitful than others. He refers to an example of an Indian song that describes the sequence of steps to build a canoe without giving any plans (blueprints) how the canoe will eventually look like.

2.4.8.2 Generativity of single patterns

The use of pattern languages guides us in generating specific designs or end products. But are general pattern languages suitable to generate all possible end products of a domain? The problem is that a complete general language is almost never possible but for very small or specific domains. Fortunately, single patterns can also be generative. A single pattern might not bring us down all along the path but it can describe an important step on the journey.

The key to generative patterns is once again to understand the forces. Rather than telling us exactly what to do step by step (like a micro-script), a generative patterns tells us how to *react* to the forces. An example is the pattern HANDS IN VIEW [6]. The pattern tells you how to react to the current situation in a generic way that accounts for the variations of the actual situation:

> "**Problem:** The skier fails to commit downhill on steeps and bumps, resulting in slides, backward falls, and "yard sales." […]
> **Solution:** Concentrate on keeping the hands in view. Bring them into sight immediately after each pole plant and turn.
> **Resulting Context:** Keeping the hands in view changes the alignment of the body from sitting timidly back and allowing the edges to skid out from under the skier. Thus, keeping the hands in view pulls the body forward and thus downhill, bringing the skier's weight over the downhill ski, forcing the edge to bite and turn" (Olson, 1995).

2.4.8.3 Generativity of pattern compounds

The pattern language of Alexander has failed in his own eyes because whenever applied it never generated the "Quality Without A Name" (Gabriel, 1998). One of the reasons might be that the application of single patterns is not enough; one needs to resolve the resulting situations. The unfolding process is more important than achieving a single static structure. In principle, the pattern language offers such a process in relating further patterns. However, the particularization of the current situation might call for more variations than a general pattern language can offer. Alexander always highlights that for each project an adapted pattern language is needed (Alexander, 1979).

Predefining explicitly a complete pattern language could be danger: people may follow the offered paths blindly – even if other patterns not explicated in the language might be more adequate. Patterns that are part of a language often have a very short discussion of the solution and its resulting context because they express them in

[6] http://c2.com/cgi/wiki?HandsInView

terms of other patterns. But if these patterns are not consulted (by ignorance or because of inadequateness) one has a less profound understanding.

The patterns in *Design Patterns* (Gamma, Helm, Johnson & Vlissides, 1995), *Pattern Oriented Software Architecture* (Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996) and many patterns of the PLoP conferences discuss the resulting contexts and consequences more deeply, mostly because they are not part of a complete language.

2.4.8.4 Generativity of path descriptions

A path is followed and created by performing a sequence of steps; it is a process in which the thing – the path itself – is generated. A particular hike along a path unfolds in the process of walking. It cannot be planned fully in advance. If a stone is on the road, a hiker needs to react and the particular course is adapted accordingly. If the hiker spots a beautiful flower or butterfly he stops at unpredictable times. A map of paths does not prescribe the exact sequence of steps but rather offers directions and constraints. The hike is volatile; each step is a transformation of the current situation. At any time the current context needs to be re-evaluated to account the local forces. A path description that is generative tells you how to proceed in the sequence of circumstances: "when you see the big oak tree, you should keep right until you find a place in the river that is not very deep, so that you can cross it barefoot".

Instead of having a geographic map with a bird's eye perspective, demotic instructions or sketches on a napkin concentrate on the actions and milestones that are important to follow the path successfully. In that respect informal descriptions are quite precise because they take into account what actually matters in the given context. The form of the description – watching out for landmarks, describing conditions for the next operation – provides context-based instructions.

2.4.9 Patterns and meaning

Patterns are based on the experiences of practitioners. They are not providing abstract design methods or general pedagogical models that leave open what to actually do. Patterns are tackling what "really" happens on the ground. General principles of design can be reflected in patterns. They are often found in the rationale section or are expressed as forces. However, isolated forces are meaningless. It is the interaction of forces in a context that makes design hard. A design has to respond to its environment to be meaningful: "We are quick to credit the success of great programmers to their methods, their programming languages, or their managers. Too often, we don't look far enough to see the *meaning* programmers add to the body of literature we call system software" (Coplien, 1998c).

2.4.9.1 Meaning in contexts

Meaning unfolds only in contexts, taking the wholeness of a situation into account: "…the whole field of meaning can be described as subject to a distinction between content and context […] content and context are two aspects that are inevitably present in any attempt to discuss the meaning of a given situation" (Bohm & Factor, 1985, p.85). A pattern only gets meaning when it fits and responds to its environment. This fitness can be linked to the survival of meaningful forms: "In my view meaning is intimately tied up with survival and natural selection. Events that happen to an organism *mean* something to that organism if those events affect its well-being or reproductive abilities" (Mitchel, 2009, p 284, highlighting in original text). Maybe the notion of survival explains how Alexander sees different degrees of life in every structure. Meaningful configurations fit to the environmental context and show a higher degree of life.

What matters – what means something – depends on the totality of a situation, "each thing *is* its total meaning – which of course must include all of its relevant context" (Bohm & Factor, 1985, p. 97). That the totality of a situation *is* the meaning rather than referring to the meaning *of* a situation is pointed out by Bortoft (1996). He uses an example of a patchwork image of black and white shapes in which a giraffe is hidden: "The experience of suddenly seeing the giraffe, for example, is the experience of seeing meaning where previously there had been only a meaningless patchwork of black and white shapes. The nonsensory wholeness or unity, which we see in the instant this patchwork becomes organized, *is* the meaning 'giraffe.' This is not the meaning *of* what is seen, but the meaning which *is* what is seen" (Bortoft, 1996, p. 52).

By organizing parts in a specific (meaningful) way we get a new meaning. The specific organization *is* the meaning. This implies that the way of looking at something provides different insights (theories). The observer becomes part of the wholeness. The same event can have different meaning for person A and person B. Likewise the same software system can have different meaning to different people. To understand the meaning of software to end-users it is important to focus on the conceptual construct of a program rather than the details of the artefact (Quillien, Rostal & West, 2009). To be meaningful these concepts have to be adapted to the actual needs of end-users. Since software designers are often not the end-users they have to build theories about how their program execution can solve the problems of end-users. These theories need to be refined and modified to account for the actual needs (Naur, 1985).

2.4.9.2 Values for humans

The Wholeness or "Quality Without a Name" that we want to achieve ultimately is for the benefits of the user: "What is this nameless quality? Most coders have had the pleasure of knowing it, when they build a particularly *satisfying* module or system, code that just 'feels right'. Of course, to define it or name it would miss the point. However, recurring themes in the pattern literature point out aspects of what such a Quality might be. One theme, close to Alexander's goals of architecture is to serve human needs" (Coplien, 1996).

But that does not necessarily refer only to end-users. The users of software patterns are not end users; the users of software patterns are software designers who are the habitants of the code: "At least one computer scientist identified the 'user' of a piece of software as the end user. This appears to make sense at first, but when you read Alexander, it is clear that a 'user' is an inhabitant – someone who lives in the thing constructed. The thing constructed is under constant repair by its habitants, and end users of software do not constantly repair the software, though some might want to" (Gabriel, 1996, p. 33). Of course, if the code is habitable it is better to maintain and it is more robust; this indirectly improves the software quality for end users. New requirements can easily be implemented and the system does not break. The idea of Alexander is to involve the users in the design process – which is "strikingly similar to the idea of participatory design, which aims to actively involve end user in all stages of the software development cycle" (Borchers, 2001, p.12).

Software design patterns address software developers but good software design has positive effects for end users. A solid architecture is aesthetically pleasant to software developers and is a key to success (Saunders, 1998) because it has the "elusive properties of good software: elegance, flexibility, extensibility, and reusability" (Vlissides, 1998). To be whole, in peace and harmony, a system has to account for the needs of the developer and of the end users.

2.5 Pattern culture

Over the years a pattern culture evolved in the software design community. This culture includes ethical values as well as processes and methods to foster high quality pattern descriptions. Alexander's attempt was to improve the world by letting people participate in the building process and create towns, neighbourhoods, buildings, gardens and rooms that are alive and that share the quality without a name. His pattern language is a literary documentation of patterns that help to generate that quality. Alexander complains about many contemporary buildings that are monuments of their architects or mass-fabricated modules rather than places where human beings feel comfortable and in harmony.

The software community in the late 1980s was in a similar situation. Although object oriented programming was a significant advance in producing software, the community "felt that object orientation had failed to live up to its 'promises'" (Coplien, 2004).

2.5.1 The need for literature about good design

One of the problems was that the academic world only provided some general ideas about good design but no specific guidance how to create elegant software: "Some of the early pattern community members, such as Ward Cunningham, were frustrated about not being able to publish small nuggets of everyday Smalltalk code that exhibited elegance and mastery of the language at mainstream conference venues. The pattern community provided a publication outlet for those ideas" (Coplien, 2004).

The goal was to collect such nuggets of wisdom in a handbook (Riehle & Züllighoven, 1996) as design literature and "spread the word about good practices in software development" (Buschmann, Henney & Schmidt, 2007, p.92). Bruce Anderson ran "Architecture Handbook" workshops at the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA) and inspired the community to build "a common body of literature". Kogut (1998) points out that "handbooks are a pervasive technique for sharing design knowledge in mature engineering disciplines. Chemical engineering has one main handbook (usually referred to as Perry’s Handbook) that has more breadth and depth than any of the fragmented set of existing software engineering handbooks/textbooks […]. Perry’ Handbook has over 1000 ‘telephone book’ style pages. It is updated about every 10 years and it is now on its 6th edition.”

Having a handbook that contains all the “standard situations” could help developers resolve recurring problems (Appleton, 2000). Most advanced disciplines like biology, physics, or chemistry have standard handbooks that contain the shared knowledge of the discipline.

In education, it seems, we are facing the same problem (Baumgartner, 2011). While there are many books on methods there is not a handbook of all the agreed methods which offers a quick access and thorough discussions. First steps into that direction are “The Teacher’s Toolkit” (Ginnis, 2002) or Flechsig’s (1996) “Kleines Handbuch Didaktischer Modelle”. However, these are volumes of single authors. Standard volumes in other disciplines are very often edited by many authors and updated frequently to take the latest research into account. It would be beneficial to have a handbook of educational and e-learning patterns that covers most of the “standard situations” of learning and teaching (Wedekind, 2011).

2.5.2 History

This section will follow some of the traits how the pattern concept made its way to the field of software design. It could give us some inspiration how the pattern idea might be further carried on in the education community.

2.5.2.1 Alexander’s foundation

The first notion of a “pattern” in the sense discussed in this thesis can be found in *Notes on the synthesis of Form* (Alexander, 1964). The focus of the book is how to synthesize forms that have a good fit, i.e. forms that balance the forces of the context and the solution. *A Pattern Language* (Alexander et al., 1977) presents a network of patterns that are such stable configurations. *The Timeless Way of Building* (Alexander, 1979) explains the concepts behind *A Pattern Language*. It introduces “The Quality Without A Name” which each of the patterns should have; the pattern language is seen as “The Gate” through which the Quality can be achieved: following the patterns is like following a well known path and its end is the Quality. The third part of *The Timeless Way Of Building* describes “The Way” in which we must enter through the “The Gate”: not blindly following patterns but differentiate them according to the specific needs, i.e. selecting the right sub-patterns and their variations. The failure of *A Pattern Language* is that people thought that patterns would just do (Gabriel, 1996). However, much more important than the patterns is the process in which the whole unfolds in piecemeal steps of symmetry-breaking and structure-preserving transformation. These insights, along with a set of 15 fundamental properties of good structure, are largely discussed in the four-volume series *The Nature of Order* (Alexander, 2002).

The pattern culture as lived by the community that actively works on new patterns in software design, human-computer-interaction, pedagogy, and many other fields, draws mainly from the first three books. The *Nature of Order* is anticipated but its meaning and implications for the community are not yet fully understood.

2.5.2.2 Patterns for software

The ideas of Alexander have been long discussed in the software community: “Alexander's early writings on structure and method have influenced designers in all realms, including computer scientists ranging from Herbert Simon to Harlan Mills” (Lea, 94). The concept of hierarchical organization of nearly-decomposable systems is very similar to Herbert Simon’s ideas on complex systems: “The study of patterns is not unique to software; indeed, the importance of patterns has long been recognized in other technical disciplines such as biology, physics, and architecture. […] Herbert Simon, in his study of complexity observed that ‘hierarchic systems are usually composed of only a few different kinds of subsystems in various combinations and arrangements’. In other words, complex systems have common patterns” (Booch, 1998).

The idea of pattern languages for software was picked up by Kent Beck and Ward Cunningham in the 1980s. Beck remembers in an interview: “When I was an undergraduate at the University of Oregon, when I was a freshman and poor, I ran across ‘The timeless way of building’ in the university bookstore and I couldn't afford it, so I read it standing up in the bookstore like going and reading a chapter, 2 chapters and then put it back on the shelve and then I'd come back and read a couple more chapters, so I was familiar with the whole Alexandrian style and attracted, frankly, to this ‘Here is the commandments’. […] When I got to Techtronics and I was working with Ward Cunningham we both had exposure to Alexander's work and we talked about patterns and how they could be applied to software development. Lots of people were talking about it, but nobody was really doing stuff about it at the time, so we started experimenting with patterns, original patterns were for user interface design" (Beck, Nilsson & Marinescu, 2008).

Beck and Cunningham published their first "language" in 1987 about user interface design in the programming language Smalltalk, consisting of five patterns: WINDOWPERTASK, FEWPANES, STANDARDPANES, NOUNSANDVERBS, SHORTMENUS. They can still be found in "Ward's Wiki"[7]. Beck and Cunningham were impressed by what end-users had designed based on the patterns. Since the patterns dealt with both software and interface design they are the "birth" for both branches of patterns. The HCI community celebrated the 20th anniversary of user interface patterns at the Usability Professionals' Association Conference 2007 in Austin, Texas, titled "Patterns: Blueprints for Usability".

2.5.2.3 The first design patterns

Some years later, Bruce Anderson run a birds-of-a-feather session (a conference slot that is open to newly proposed topics, not unlike an open space or bar camp) at the 1990 ECOOP/OOPSLA in Ottawa with the topic "Towards an Architecture Handbook". At this workshop Erich Gamma and Richard Helm were present (Coplien, 1996).

Gamma was working on his PhD thesis about recurring structures – patterns – in object-oriented design. It was published in 1991. In the same year he and Helm were "sitting on a rooftop in Zurich on a sweltering summer's day, put together the very humble beginning of the catalogue of patterns that would eventually become Design Patterns" (Coplien, 1996). A short time later the "Towards an Architecture Handbook" workshop was repeated at OOPSLA 1991; there Gamma, Helm, Johnson and Vlissides – later known as the "Gang of Four" – all met[8]. In 1995 they released their book "Design Patterns" (a pre-version was available at OOPSLA 1994) which is fairly one of the most influential books in software design if not computer science. Two other books had followed the pattern idea at the same time. James Coplien (1992) had started to collect language specific C++ patterns and published them as "idioms" in his book *Advanced C++ Programming Styles and Idioms*. Around 1991 a group of European Software developers, too, started to work on patterns. Their book *Pattern-Oriented Software Architect* was published in 1996 (Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996). It was followed by four more volumes; it also started a series in software design patterns published by Wiley. This series includes books about interaction design as well as computer mediated interaction.

2.5.2.4 Foundation of Hillside

In 1993, the Hillside group first met, "a group of seven people met in Colorado to evaluate how patterns might address the problem of the object community: Ken Auer, Grady Booch, Ralph Johnson, Hal Hildebrand, Kent Beck, Ward Cunningham, and Jim Coplien" (Coplien, 2004). This group came together again in 1994. This time Richard Gabriel was there as well and he proposed to discuss patterns in specific way: during a Writers' Workshop.

In August 1994 the first Pattern Languages of Programs (PLoP) conference was held in Allerton Park, Illinois. It was run by the Hillside Group who still organizes PLoP conferences all over the world. Its mission statement starts with: "The mission of the Hillside Group is to improve the quality of life of everyone who uses, builds, and encounters software systems - users, developers, managers, owners, educators, students, and society as a whole."[9]

While the Hillside Group certainly focuses on software patterns, the PLoP conferences have always been open for other topics as well; many pedagogical, human computer interaction, e-learning, and management patterns have been discussed at PLoPs. Other pattern conferences include EuroPLoP, AsianPLoP, ScrumPLoP, VikingPLoP, SugarLoafPLoP, and ChiliPLoP. The EuroPLoP is currently the one with the most attendees and it is sponsored by Hillside Europe, a German non-commercial association.

2.5.3 A community of trust

The PLoP conferences are very different to other conferences. The main venues are the Writers' Workshop. These sessions last 60 to 90 minutes and the authors of a paper do not present it but receive feedback from other people in the workshop. The months before the conference are used for a shepherding process. During shepherding an experienced pattern author provides several rounds of feedback to the authors of a paper in order

[7] http://c2.com/cgi/wiki
[8] http://c2.com/cgi/wiki?HistoryOfPatterns
[9] http://hillside.net/

to improve it step by step. Both Writers' Workshops and Shepherding are not about defending the author's work but to explicitly improve it. Since the papers are criticized there is a need for trust between the members of the community.

During the conference there are many activities to build trust and strengthen the community. There are a number of ways in which new members are welcomed and apprenticed. The (US) PLoP runs a Bootcamp in advance of the conference in which the concepts of patterns are elaborated; part of the Bootcamp is a writing activity to develop a simple pattern. Each conference runs a demo Writers' Workshop to make everyone familiar with the format. Weak paper submissions are not simply rejected; some PLoPs have on-site shepherding where an experienced community member works with an author on her or his paper during the conference. The EuroPLoP starts with a "Newcomers introduction". There is also a shepherding workshop to grow new shepherds. At EuroPLoP 2010 there was a full afternoon workshop on how to write good patterns. During all of these activities one learns a lot about what a pattern is and about the diversity of views. All PLoP conferences have dedicated slots for games. A game master organizes this activity during which the participants get to know each other. EuroPLoP also has one art session in which all participants of the conference work together on one artefact. The conference dinners offer a lot of opportunities for heated discussions and to make friends. The location of PLoP conferences is often distant from larger cities and people keep together rather than spread out to different hotels. The EuroPLoP has been held for 15 years at Kloster Irsee. People come together every evening for sharing their ideas or run birds of the feathers – spontaneously initiated discussions about a proposed topic. The (US) PLoP has been in Allerton Park for more than 12 years, a beautiful place that offered calm and time to focus on patterns. Many people who have attended a PLoP consider it as an outstanding experience because it is very different to mainstream conferences[10]. It is much more effective because one gets a lot of feedback for one's own work and one has to deeply consider the work of others in order to provide feedback to them. The conferences are a lot of fun but also much more work than other conferences.

2.5.4 Values

The Hillside Group designed the pattern conferences differently than major academic conferences to establish certain values and ethics within the community. Coplien (1996) names the following values in the pattern culture:

- The Quality without a name: the documented patterns should have this quality.
- Real stuff: patterns should be based on actual designs not pure ideas.
- Constraints are liberating: a patterns constraints the design decisions to avoid failures and foster creativity
- Participative architecture: pattern languages can help end-users expressing their needs
- Dignity for programmers: not a separation of designers and programmers
- Aggressive Disregard for Originality: patterns should capture long proven ideas
- The Human Element: software serves a human need, patterns should address the human interest
- Aesthetics: Good design is appealing for programmers and they feel more comfortable "in" the code
- Interdisciplinary scope: patterns balance the needs of developers, interdisciplinary teams, organizations and customers
- Ethics: the community shares some ethics, e.g. protection of intellectual property or avoidance of making patterns a hype

Some of the values have been discussed in the previous sections about the quality of patterns. The focus of the next sections will be on how the community lives those values and which organs it has created to maintain the value system.

2.5.4.1 Patterns based on empirical data

For Alexander, patterns can be induced from good examples but also deduced from lessons learnt or pure reason. The pattern community, tired of too abstract academic ideas that do not work in the field, values induction of patterns based on existing designs that have shown to work. The "aggressive disregard for originality" means that patterns should capture "proven" ideas rather than new ideas. They are based on "real stuff" (Coplien, 1996).

[10] Two reports on the conference can be found at: http://www.peter.baumgartner.name/Members/baumgartner/news/ueber-schafe-und-schafhirten---zur-europlop-2011

In their book the Gang of Four reflects: “It’s possible to argue that this book hasn’t accomplished much. After all, it doesn’t present any algorithm or programming technique that haven’t been used before” (Gamma, Helm, Johnson & Vlissides, 1995, p. 351). However, there is a misconception about the relation between original patterns and original solutions. A pattern that describes a solution that has occurred a million times is still an original work if it has not been mined before as we will see in section 3.4.2.1.

The notion of “real stuff” and “disregard of originality” was addressing a world of computer science that was dominated by mathematics and purely theoretical ideas. The aim of the pattern community was to find patterns for which empirical evidence was available. The “Rule of Three” (i.e., a pattern should be based on at least three proven designs) was not only a container of evidence. It was also a way to keep academics out of the community that saw patterns as a cheap way to publish their new (untested) ideas. Indeed the value of the pattern idea is weakened again and again by people who use the pattern form but do not present patterns based on real instances in the world (Coplien, 2004).

The power of patterns lies in the observation of good designs, “many people will agree that a great architect’s creative power, his capacity to make something beautiful, lies in his capacity to observe correctly, and deeply. […] In this sense, then, a pattern language which is deep is a collection of patterns which correspond to profound observation about what makes a building beautiful” (Alexander, 1979, p. 218). Of course, the observer starts to reason about the sources of that quality. The reasons are given usually in the forces sections. This understanding is obviously more than could be observed directly. So, patterns are more than just recurrent structures and “patterns in fact represented a strong return to scientific empiricism that could be contrasted with the ad hoc reasoning that had guided much of software until that time” (Coplien, 2004).

2.5.4.2 A culture of sharing

Patterns are way to share knowledge. In particular, the tacit knowledge in the heads of experts becomes unfolded in an explicate representation – the pattern description. The pattern community has created a way of sharing this knowledge in a literary format, discuss it on conferences and publish it in papers, journals and books. This sharing of knowledge transcends cultures, corporate boundaries and organizations (Coplien, 2004) and is not unlike open-source sharing or the sharing of knowledge in science and arts.

The pattern community consciously shares knowledge and cultivates a culture of giving (Gabriel, 2002). Not only is the nugget of wisdom captured in a pattern a gift to other experts and beginners. The feedback provided for the patterns during and before the pattern conferences is also seen as a gift. At the (US) PLoPs this spirit of giving is symbolized in the tradition of bringing small gifts for other participants to the conferences.

2.5.4.3 Writing quality

To effectively share the knowledge about patterns, the community pays a lot of attention to the literary quality of a pattern description. In fact, the format of the pattern conferences has established Writer’s Workshops and a Shepherding Process (see below) that are designed to improve the quality of the written words rather than the content. It is trusted that the author is an expert of her or his pattern and knows it very well. The focus is to support the author to bring that knowledge to paper in a way that it is understood by others. Of course, in this process style and content of a pattern change – very often shepherds and other people in a Writers’ Workshop can help the author to clarify statements and facts and pull out more interesting information.

Many people invoke the metaphor of a story or a play (Rising, 1998) and point out that patterns are not just about facts but should tell a story (Appleton, 2000). As in a play the forces are creating a tension and the solution is resolving the conflict – a happy end.

Doug Lea has compiled a checklist about writing good patterns. A pattern describes “a single kind of problem, the solution as a constructable, design steps or rules for constructing the solution, the forces leading to the solution, evidence that the solution optimally resolves forces, details that are allowed to vary, and those that are not, at least one actual instance of use, evidence of generality across different instances, variants and subpatterns, other patterns that it relies upon, other patterns that rely upon this pattern, relations to other patterns with similar contexts, problems, or solutions.”[11]. There is also a pattern language for writing patterns that provides guidance in writing patterns (Meszaros & Doble, 1997). In addition to these patterns Harrison (2006b) collected further patterns for advanced pattern writing. Wellhausen and Fießer (2011) address first time pattern authors and reflect

[11] http://hillside.net/index.php/pattern-writing-checklist

on their experience in writing patterns. They point out that patterns are not written in the sequence they are read; very often the writing starts with the solution, followed by the problem, consequences, forces and context. Vlissides (1996) identifies seven habits of successful pattern writers: taking time to reflect, adhering to a structure, being concrete early, keeping patterns distinct and complementary, presenting effectively, and collecting and incorporating feedback.

2.5.5 Shepherding

Before a pattern paper is accepted for a Writers' Workshop, it is always shepherded by an experienced pattern author (Harrison, 2006a). The shepherd builds up a personal relation to the authors and provides iterative feedback to the pattern paper in about three rounds. The non-anonymous peer review creates a trusted relationship. The first iteration of feedback usually provides feedback for the big picture, i.e. general suggestions for improvement. To not overwhelm the authors with suggestions, the first round often starts with "Half a Loaf": "Send the author small bites of comments. Start with the biggest issues, and work to the most trivial.
If you run out of time but haven't assembled all the comments you wish to make, send what you have. The author can start on whatever comments you send. Half a loaf is better than none" (Harrison, 2006a). The last iteration often gets more detailed. Some shepherd even offer grammar correction to non-native speakers.

This reflection on the patterns and the chosen writing style enhances the documented pattern but it has also side effects on the process of pattern mining. The core of the pattern becomes clearer on every reflection and many pattern authors have reported that they learned much about the pattern while they were writing it down (Rising, 1998).

It is important to note that a shepherd is just providing comments and suggestions. The paper is owned by the authors. It is up to them what to make of the suggestions. A shepherd is not supposed to prescribe any changes; s/he can only provide recommendations. A good shepherd also avoids providing the answers to the questions s/he has. The shepherd should stimulate the authors and not take on their job. S/he should rather ask: what are the forces than just name more forces. The goal is to make the sheep think and pull out more of their tacit knowledge.

The volunteer work of shepherding is also understood as a gift to the authors and to the community. It is a good habit of sheeps to thank the shepherd in an acknowledgement section of the paper. The efforts of shepherds are also acknowledged by a shepherding award that is given out at some PLoP conferences.

The shepherding process is one of the most important organs of the pattern community. The process is so important that the activity is monitored silently by members of the program committee to ensure that both sheep and shepherd act appropriately. Very often, books about patterns are also shepherded by other members of the community.

2.5.6 Writers' Workshops

The idea of Writers' Workshops was brought to the pattern community by Richard "Dick" Gabriel, a software developer and poet. In one of the first Hillside meetings he proposed the format that he knew from poetry conferences: "Richard Gabriel, himself a poet, taught us how poetry is reviewed at poetry conferences. We adapted that approach and call these sessions Writers' Workshops" (Coplien, 1996).

A Writers' Workshop is an effective format to review and evaluate a pattern description and point out suggestions for improvements in a friendly environment. During a PLoP conference the members of a workshop group stay within their group for the entire conference; one of the ideas is to build trust between the members. Each session is used to discuss one paper that contains pattern descriptions. Each participant has thoroughly read the paper and prepared comments for it, in order to highlight strength, weaknesses and suggestions for improvement.

One curiosity of the workshop is that the author becomes a "fly-on-the-wall", that is s/he only listens to the discussion of the workshop members about content and style, but cannot defend the paper. The idea is that a documented pattern has to stand on its own and must be understandable for readers without further explanation of the author. The feedback gathered is very valuable. The authors learn which parts of their paper work or have to be improved. They also learn whether their own concepts and ideas are shared by other people.

The typical structure of a Writer's Workshop is as follows (Gabriel, 2002):

Authors read selection
The authors read a short paragraph from their paper. The idea is to hear the author's voice – to remember everyone that behind the paper there is a human being. It also is a "formal" sign that the workshop has started and people start focussing on the paper.

Fly on the Wall
For the next sections the authors become flies on the wall. That means they are not supposed to contribute to the discussion but just listen how their paper is perceived. The paper should explain itself without the help of the author, therefore authors are not allowed to defend or explain it.

Volunteer Summarizes the Work
One or two persons in the workshop summarize the paper: What is it about? Who is the target audience? Which patterns can be found in the paper? To which other work does it relate? This summary should give the author an idea how her or his work is perceived. S/he can hear whether readers get it and understand its intent.

Positive Feedback First
The group first discusses the positive aspects of the paper: what has worked and what should be kept as it is? Positive feedback can be given for the content, the pattern format, the writing style, illustrations, the structure, etc. To start with positive feedback gives the author a feeling of safety – not everything is bad – and shows that others value the work. Giving positive feedback for a weak paper is sometimes a challenge. Amazingly enough, even the worst paper has something good to say about. An old joke at PLoP conferences is: "I liked the whitespace". Of course, in a real workshop the art is to highlight positive elements of the paper that could be a model to improve the rest of the paper.

Suggestions for Improvement
This is the heart of every Writers' Workshop and takes most of the discussion time. The workshop group provides constructive feedback. That is they point out what they did not like about the paper in a non-offending way and provide specific suggestions what could be done to improve it. One should not hear "this paragraph is ugly" but rather "this paragraph could be improved by doing this". To not offend or attack the author personally it is custom to speak of "the author", i.e. one says "the author should use coherent language in this paragraph" rather than "Chris messes up with the language". In the discussion, the author will sometimes hear different opinions about the same matter. However, if the group agrees on some positive or weak aspects of a paper there is no need to waste time and repeat the same comment again and again. There is a common shortcut to signalize agreement: people say "gush" if they agree. The verbal signal of agreement is important because a fly on the wall cannot see nodding or shaking heads. The responsibility to play by the rules (e.g. not offend the authors) is taken by each reviewer. The workshop moderator should remind them if they disobey the rules. S/he is also in charge to move on in the discussion. If a point is made there is no need to discuss it again and again.

Positive Closure
The workshop started with positive feedback and it ends with positive feedback (a "sandwich") to leave the author with a positive feeling. One of the reviewers highlights one outstanding positive aspect of the paper: a new insight, a particular well written paragraph or the value of the work.

Author Asks for Clarification
After the discussion the fly on the wall returns back to the discussion circle. Since s/he was mute during the discussion, s/he can now ask questions of clarification. This is not an opportunity to defend or explain the paper. Such attempts are usually cut off immediately by the moderator. The author is allowed to ask for clarification about the comments.

Thank the Author
The workshop ends with applause for the author. This is a way to thank the author for the gift, the valuable work in the paper, and displays gratitude for the author. It also formally ends the workshop session.

There are different variations and every moderator develops her or his own style. For example, at the EuroPLoP conference the "Fly on the Wall" actually moves out of the circle and does not face the group any longer; at

PLoPs in the US, the fly usually remains in the circle but is not supposed to speak. Some moderators, however, bring the fly back into the discussion every now and then. Another style first discusses the positive aspects and improvements of the pattern formats and then digs into each pattern, following the same rhythm. Another variant is to let the participants comment one after another, to ensure that everybody is involved. Gabriel, who has introduced the sequence to the community, does not follow it any longer but leads a more open discussion as it is common in Writers' Workshops for poetry.

A written pattern becomes empirical vulnerable in a Writer's Workshop and it is an established method to find weaknesses in the written pattern to further improve it. Making patterns explicit and external accessible allows other people to adapt to this knowledge, discuss it, reuse it or improve it. The benefit of the method is to gather many different perspectives, reveal factual errors, grammatical weaknesses, illogical arguments, inconsistencies, and cultural faux-pas.

A more elaborate discussion about the elements and backgrounds of a Writers' Workshop can be found in "A Pattern Language for Writers' Workshops (Coplien, 2000) and "Writers' Workshops & the Work of Making Things" (Gabriel, 2002). Allan Kelly also provides a flowchart that illustrates the flow of activities[12].

People who have experienced Writers' Workshops often start to employ them in their own academic environment. Reinhard Bauer and Gabi Reimann (2010) have started to experiment with Writers' Workshops to foster scientific writing skills. Schmolitzky & Schümmer (2011) have written a pattern about Writers' Workshops for providing feedback to students.

2.5.7 Publication process

The publication process of patterns is in many respects different to traditional scientific publications. The Writers' Workshop and the Shepherding process in particular make the publication process different. The final paper version is not due to the conference and automatically ends in the proceedings; rather, the authors continue to work on the paper after the Writers' Workshop and submit the final version later. The ideal publication process is as follows (Quibeldey-Cirkel, 1999a):

- Pattern Mining: Practitioners or researchers find nuggets of wisdom, e.g. patterns of good solutions
- Pattern Writing: Authors describe the patterns in one of the pattern formats
- Shepherding: The pattern is improved in several rounds of shepherding; often these improvements are not limited to writing style but bring more substance to the pattern
- Writers' Workshop: The patterns are discussed by several reviewers in Writers' Workshop, for example on a PLoP conference
- Final submission: The comments from the Writers' Workshops are used to improve the final version. Many reviewers give their printouts of the paper with written notes to the authors.
- Publication: The final patterns are published in conference proceeding, books, websites, or journals

The patterns of the first PLoPs were published by Addison Wesley in volumes. The EuroPLoP used to publish the proceedings of the previous conference at the next EuroPLoP. Both conference now publish the proceedings as ACM online publications. Many patterns have been published in Wikis – the first Wiki was written to collect patterns (Portland Pattern Repository). There is also a Springer journal: "LNCS Transactions on Pattern Languages of Programming". The journal follows the standard peer reviewing process of scientific publication but has an additional requirement that each submission had to be discussed in a Writers' Workshop on a PLoP conference before.

2.6 The Nature of Order[13]

The spirit of the pattern community largely builds on Alexander's early works. But there is an important four volume book, *The Nature of Order*, that takes new insights of Alexander into account. A short introduction to the volume can be found in "*A Delight's Muse*" (Quillien, 2008). Patterns capture the invariant of several exemplary structures. For example, MAIN ENTRANCE (Alexander et al., 1977) describes general design decisions for main

[12] http://www.allankelly.net/patterns/workshopleader.html
[13] Chapters 2.6.1-2.6.4 have been published in a modified version as "Patterns as an analysis framework to document and foster excellent e-learning designs" (Kohls, 2011b).

entrances that have the "Quality Without A Name". *The Nature of Order* on the other hand proposes more abstract and universal properties of good design. Alexander claims that there are about 15 fundamental properties that can be found universally in good and well ordered objects. They apply to architecture as well as animals, plants, and all organic structures. As such they could potentially lead to general ways to achieve more lively forms. Put it differently, the properties are supposed to provide insight to what makes a form a good and beautiful gestalt. Since these new ideas have been discussed on pattern conferences in the last years, we will introduce the 15 properties and discuss their different interpretations in separate sections. There is a strong connection to gestalt theory that is reflected in Alexander's terminology (e.g. "centres", "wholes" and "wholeness"). To better understand the conceptual roots we will first relate gestalt theory to Alexander's work before each property is tackled. The chapter closes with a critical discussion of the 15 properties and their meaning for subsequent parts of this thesis. Unlike the previous sections, there is not much evidence and empirical support for the claims. As the discussion still goes on, the reader needs to be aware that this subsection is often speculative and only represents first impressions.

2.6.1 Centres

In his latest work Alexander introduces the term "centers" for wholes: „I use the word center to identify an organized zone of space – that is to say, a distinct set of points in space, which, because of its organization, because of its internal coherence, and because of its relation to its context, exhibits centeredness, forms a local zone of relative centeredness with respect to the other parts in space" (Alexander 2002a, p. 84).

To consider wholes rather as centres is a shift of thinking that makes the connection between sub-wholes (centres) and larger wholes (again centres) more explicit and tries to avoid the question where a whole accurately starts and stops. If we look at a lecture it is hard to say when it does start exactly: When students enter the lecture hall or an online room? When the docent enters the room? When the docent starts speaking? Similarly we can wonder at which point it ends. When the docent stops speaking? When the students have left the building? Or when they stop talking in smaller groups about what was presented in the lecture? There is, it seems, a centeredness making certain activities in time and space more belonging to a lecture (e.g. the actual delivery) whereas other activities are less connected to the centre of the lecture but still belong to it (such as immediate discussions "afterwards"). The terminology of wholes and sub-wholes or parts and sub-parts incorrectly implies that there is an accurate boundary. Therefore it is more appropriate to speak of centres: "The same is true for window, door, wall, or arch. None of them can be exactly bounded. They are all entities, which have a fuzzy edge, and whose existence lies mainly in the fact that they exist as centres in the portion of the world which they inhabit" (Alexander 2002a, p. 85).

To speak of centres stresses the point that a form depends on its surrounding, that a pattern cannot be seen in isolation but gets its meaning from the context in that it is used. That centres are to some extent what Alexander had used to call patterns is mentioned only in a marginal note in *The Nature of Order*: "The entities we called patterns were – albeit in an early formulation – somewhat similar to the entities I now call centers" (Alexander 2002b, p. 344). Coplien (Coplien & Appleton, 1997) interprets patterns as stereotypical centers.

2.6.2 Centres and whole parts

2.6.2.1 Gestalt theory

While the number of inner relations seems to be very technical and formal, Alexander (1964) argues that the partition of a whole into sub-wholes can be naturally (intuitively) grasped according to Gestalt psychology: "The question is, how are these separable structural components of a problem to be recognized? We face this kind of task everyday, constantly, even when we see nothing more complicated than a pair of oranges on a table side by side. In seeing two oranges laying side by side, and not one and a half oranges lying next to half an orange, we have recognized the structural components correctly. (Correctly, of course, because while we can pick either orange up and leave the other where it is, we cannot pick up 1 1/2 , and leave ½ orange lying there.). Köhler and Wertheimer drew attention to the fact that even an apparently simple cognitive act like this, in fact demands a very complex perceptual operation" (Alexander, 1964, p. 117).

In the same way we can intuitively grasp wholes and self-contained parts of a larger whole. For example, consider the two parts of a car in figure 19.

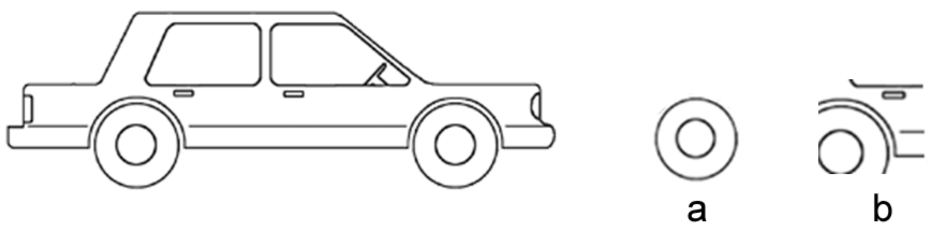


Figure 19. Parts of a car.
While both parts recur with the same frequency in the world, only one is perceived as a whole part.

If we take parting just as an arbitrary fragmentation then both parts are equal in that they are parts of the whole structure. But the partition of (b) makes no sense (it has no meaning to us). The partition of (a) on the other hand seems to be meaningful. How comes that we can see the wheel in isolation when we have argued before that a form fully depends on the whole? Actually the form of the wheel is seen as a wheel because we do have the context of a car. Putting the very same form in a different context, we can see that it is no longer perceived as a wheel (return to figure 11 and you will find that the form can be perceived as an eye).

The perception of such wholes is the subject tackled by gestalt psychologists, and Alexander's ideas build to a large extent on gestalt theory:

"We may think of these entities as parts (as they may sometimes seem to us) or as local wholes or sub-wholes. […] these parts and entities are rarely pre-existing. They are more often themselves created by the wholeness. This apparent paradox […] is a fundamental issue in the nature of wholeness: the wholeness is made of parts; the parts are created by the wholeness. […] Thus a human head, or ear, or finger is a discernible whole. It is also, both visually and functionally, a center" (Alexander, 2002a, pp. 83-84).

2.6.2.2 Roots of gestalt theory

That the whole is more than the sum of its parts is a central concept in Gestalt theory and in the pattern approach as well (Coplien & Appleton, 1997; Buschmann, Henney & Schmidt, 2007). It has its origin in Aristotle's work (Metaphysica: Aristotle, 1957). In 1890 Ehrenfels published a paper titled "On Gestalt Qualities" (Ehrenfels, 1890) which became the founding document for Gestalt theory. It observed that the quality of a melody is not derived from hearing single isolated tones. In addition, a melody can be transposed into a different scale but can still be recognized as the same melody. In the same way different squares, varying in size or colour, are recognized as squares because of the "squareness". The structural quality is not due to the commonality of parts which can be very different but due to a Gestaltqualität that is independent of the qualities of the separate elements. A particular melody does not possess its form quality (gestalt) by the qualities of the separate notes but due to their relations (King & Wertheimer, 2005). This view was a direct challenge to the analytical, reductionist methodology performed by structuralists who fragmented the world into isolated elements and features that could then be tackled independently.

2.6.2.3 Goethe's Morphology

The term "gestalt" is taken from Goethe's morphology and "referred to the self-actualizing wholeness of organic forms. Goethe considered all advanced structures of a plant or an animal to be transformations from a single fundamental organ. He accounted for similarities among the members of a species by formal laws of (self-) organization, ultimately derived from an ideal type he called an Urbild, and attributed the differences to environmental effects" (Ash, 1995, p. 85).

While Ehrenfels concentrated on the emergence of Gestaltqualitäten, making wholes different from their parts, Wertheimer claimed that specifiable functional relations exist, "which decide what will appear to be or function as a whole and as parts" (Ash, 1995, p.121), making certain divisions of things more likely. The general approach to Gestalt theory has been outlined in (Wertheimer, 1912) but a formal summary of what Wertheimer exposed about Gestalt theory in his lectures on epistemological problems is found in a footnote of von Wartensleben (1913): "…the contents of our consciousness are mostly not summative, but constitute of a particular characteristic 'togetherness,' that is, a segregated structure, often 'comprehended from an inner center' […] The epistemological process…is very often a process of 'centering,' that is, of 'structuring' or of grasping that particular aspect which provides the key to an orderly whole, of 'unifying' the particular individual parts that happen to be present; what results is that a structured unit emerges as whole due to, and through, the power of this centering. […] This structure that results from the epistemological process depends to some extent not only on the object, but also on the observer…" (translation from King & Wertheimer, 2005, p. 106; an alternative translation can be found in Ash, 1995, p. 123).

2.6.2.4 Gestalt laws

The question of interest then was what kind of "internal laws" govern the grouping of such gestalt structures? Up until now gestalt psychologists have identified more than 100 different laws that are coherent with the gestalt ideas (Chang, Dempsey, Dooley, Laurence, Touvinen & Juhani 2002). These laws are rather heuristics than real laws (Goldstein, 2009) and, interestingly enough, only a very small subset is usually found in text books. Not all laws are limited to visual perceptions. For example, Gestaltqualitäten emerge in auditory sensation as well.

Wertheimer (1938) demonstrated several principles by which groups of stimuli organize themselves in perception. What is common to all of these principles is that "perception tended, wherever possible, towards simplicity, symmetry and wholeness" (Gordon, 1989). It is indeed interesting to notice that the grouping of objects into figures is in all cases based on a certain kind of similarity. Similar forms, similar positions, similar colours, similar motion ("law of common fate"), similarity of slopes ("law of continuation") etc. let us group objects into a unit.

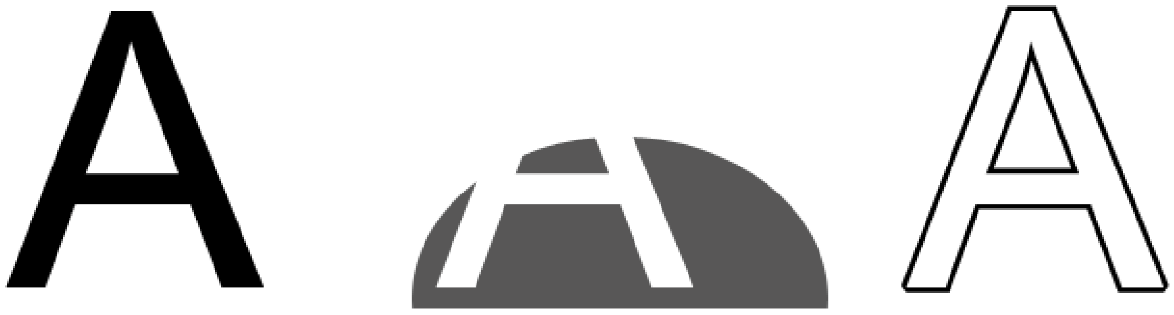


Figure 20. The "A" is perceived because of the contrast.

If the grouping is done by similarity, then the distinction between objects is based on contrast of features, accordingly. This is the basic principle of separating figure and ground. If we think of a black letter "A" on a white background, the reason that we can see an "A" is that the "A" is contrasted in the dimension of color. We can see how contrast helps the coming-into-being: If we put a white "A" on a white background there would be no way to see the "A". Hence, the background supports the "A" and provides positive space. Another way to let the "A" standout of the background is to draw an outline around the "A" in a different color. This outline perfectly mimics the form of the "A" but creates a boundary to the ground.

2.6.3 Fifteen properties that contribute to the whole

According to Alexander, the degrees of life of patterns, or centres, are themselves based on centres that are alive. But what makes a centre alive? Alexander has analyzed thousands of buildings, in real and on photographs, and he induced 15 fundamental properties – many of them related to gestalt laws - that make any structure in the world more alive than others that lack the properties. We will first discuss each of the 15 properties along with different interpretations of them and conclude with some final remarks.

We will draw from interpretations for the design of software (Coplien & Appleton, 1997; Coplien, 1998a; Wirfs-Brock, 2008; Wirfs-Brock, 2010; Waguespack, 2010), for computer mediated interaction (Schümmer, 2005a; Schümmer & Lukosch, 2007), poetry (Gabriel, 2008b) and education (Bauer & Baumgartner, 2011). The ideas about the meaning of the 15 properties for education have been discussed on a research workshop in Vienna, hosted by Peter Baumgartner and co-hosted by Christian Kohls. The discussion was recorded and the examples draw from a commented protocol (Forschungswerkstatt, 2009). There are also three works that discuss *The Nature of Order* and give summaries and explanations of the fifteen properties: *Delight's Muse* (Quillien, 2008), *Mustertheorie* (Leitner, 2007) and *Thriving Systems Theory and Metaphor-Driven Modelling* (Waguespack, 2010). Each of he next sections will start with a diagram scanned from *The Nature Of Order* and a brief summary quoted from Alexander (2002a, p.239ff).

2.6.3.1 Levels of scale

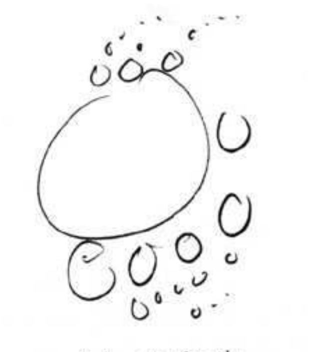


Figure 21. Levels of scale

"LEVELS OF SCALE is the way that a strong center is made stronger partly by smaller strong centers contained in it, and partly by its larger strong centers which contain it" (Alexander , 2002a).

The essence of levels of scale seems to be the appropriate ratio (the right proportion) between centres. That things should be in their right measure was already known to the Greeks (Ghyka, 1977). What matters is the ratio between the centres; for example the right ratio between group size and room size, the right ratio between content and available time, the right ratio between chapters in a book (Leitner, 2007) or on a conference presentation. There should also be a balanced ratio between lessons and pauses: a three hour lesson followed by a 2 minutes pause is inappropriate; the same duration of lessons and pauses seems inappropriate as well (Forschungswerkstatt, 2009).

Software can be viewed at different levels – conceptual, specification, and implementation (Wirfs-Broock, 2010). The ratio between the levels should be proper, "no object, method, or service should be too big" (Wirfs-Broock, 2010). In the domain of computer mediated-interaction, the levels vary in group size: single user actions, small groups, or large communities (Schümmer & Lukosch, 2007).

But what is an appropriate ratio? Is it the "Golden Ratio" as Appleton (2000) suggests? Alexander gives the rule of thumb that a ratio of 1:3 is the right level of scale, which is close to e=2,781 as highlighted by Coplien & Appleton (1997). In another paper Coplien extends the range to a factor of 4 to a factor of 10 (Coplien, 1998a). Leitner (2007) mentions a factor of 2 to 4. It is hard to see where these values come from, given that they are all interpretations of Alexander's work.

A counter argument for such numeric factors seems to be the following example: a tree on a meadow can be in perfect harmony with the flowers; yet the level of scale between trees and flowers can easily be 100.

Level of scales is rather a property that suggests no abrupt contrasts between centres. A skyscraper in the middle of town houses does not have the right level of scale because the change of size is not smooth.

2.6.3.2 Strong centres

2. Strong centers

Figure 22. Strong centers

"Strong centers defines the way that a strong center requires a special field-like effect, created by other centers, as the primary source of its strength" (Alexander , 2002a).

Strong centres seem to be centres that have their own identity. They are reinforced by other centres just as patterns are made of other patterns. In Coplien's (1998a) view design patterns help to generate strong centres, for example by decoupling tasks and encapsulate them in classes where each class is a (strong) centre. Wirfs-Brock (2010) identifies the following strong centres in software design: well-defined roles and patterns of interactions, control centres, domain models, domains and relationships between them, abstract classes and inheritance hierarchies. Waguespack (2010) interprets strong centers as cohesions or design choices that support one another in a collective.

It seems that strong centres, in opposition to "ordinary" centres, play a more central role than the smaller centres. This is also the case for a pattern of a higher level in the hierarchy of a language. A strong centre has more relations to more other centres; while some centres may just work with two or three other centres, a strong centre works with many other centres. A strong centre can have a large impact on a system design, or an educational scenario. Examples are "important events for all group members, such as kick-off meetings" (Schümmer & Lukosch, 2007), the captain of sports team or the teacher of a class (Leitner, 2007), a place in a poem that attracts attention such as "stress/unstress; sound; unit of syntax; rhyme; repeated words and sound; line; first word in a line; last word in a line; stanza; image; metaphor and other figures; title; the poem itself; historical or political elements; revealed metaphysics, meaning" (Gabriel, 2008b).

In a strong centre, the entities involved are also closer related to each other than to other entities. This is analogue to the gestalt law of proximity in which the entities belong to a group because they are closer together, i.e. more related by spatial similarity. So we can interpret a strong centre as a clear, structurally related form unit that calls attraction and shines out. Situations of confusion lack a strong centre (e.g. when students do not know whether they should follow the teacher or work on their own); a big ball of mud in software code (Foote & Yoder, 1997) is another example for missing strong centres.

That centredness is a universal concept is highlighted by (Volk, 1995, p.102): „This center-stabilized pattern is so basic to so many systems from physics to psychology that I regard it as a meta-pattern."

2.6.3.3 Boundaries

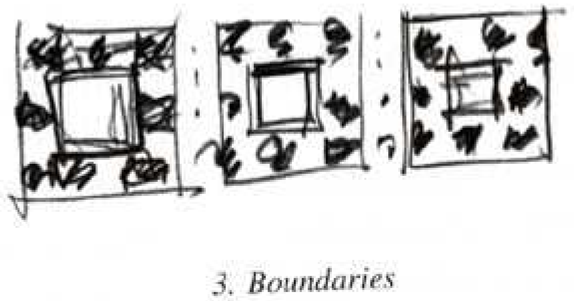


Figure 23. Boundaries

"Boundaries is the way in which the field-like effect of a center is strengthened by the creation of a ring-like center, made of smaller centers which surround and intensify the first. The boundary also unites the center with the centers beyond it, thus strengthening it further" (Alexander, 2002a).

Boundaries mark connections between different centres by both separating and not-separating the centres. The effect of boundaries is the grouping of the elements within a boundary. Its form is similar to the outlines of the centres; at the same time it contrasts each of the centres to clarify the separation.

Since a boundary is separating and connecting, it should not be interpreted as a safety guard, although a boundary could "shield group members from other group members and thus support better interaction in the group" (Schümmer & Lukosch, 2007). Seeing a boundary as a protective border around national states or a security wall around sport events (Leitner, 2007) may be inappropriate. To bound means rather grouping together than keeping out.

Yet boundaries are the place where one centre stops to be and another centre starts to be; it bounds two centres but also divides them. Such a divide is due to contrast of criteria, such as red/blue, left/right. We cannot always clearly perceive the criteria or contrasts. For example, a red circle on a red background cannot be seen. An outline in a different colour could clarify figure and ground (see also figure 20). On a sports field, it may not always be clear who belongs to which team unless there is a clear boundary (e.g. different colours of shirts).

In software code it would be hard to understand which lines of code belong together if they would not be separated by method headers, interfaces, or blocks. Waguespack (2010) maps boundaries to the concept of encapsulation.

In teaching activities it is often not obvious when one phase ends and another begins (Forschungswerkstatt, 2009). For example, if students first collect items in a brainstorming and then find categories as headlines for the items, the activity looks similar (i.e., students call out words). However, there is a strong difference between collecting items for a topic and then find classifications for the items. A clear announcement "We now start to categorize" makes the change of method and intent obvious in spite of the superficially similar activity. Other examples for boundaries are change of rooms on a conference, or pauses between lessons.

2.6.3.4 Alternating repetition

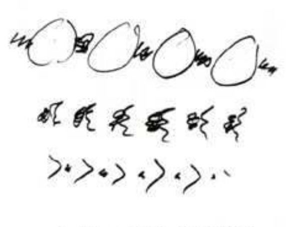


Figure 24. Alternating repetition

"Alternating repetition is the way in which centers are strengthened when they repeat, by the insertion of other centers between the repeating ones" (Alexander, 2002).

This property seems to be a combination of three others: echoes, roughness and contrast. There are repetitions of forms (echoes) that are alternating by contrasting features; as a result they highlight each other. Alternating repetitions are maximizing centres: centres are created by repetition and being highlighted by contrasts.

In time, alternating repetition could be considered as rhythm: "Alternating Repetition comes into play when different phases in the group process require a different level of involvement. Group members will follow a rhythm of participation and passivity, which is then used to participate in other social contexts" (Schümmer & Lukosch, 2007). It is a sequence of centres that repeat but every repetition is differentiated to its local surrounding: "strong centers repeated with alternating centers; not simple repeating: pattern with variation" (Gabriel, 2008b). In educational scenarios we find many alternating repetitions: a script or textbook has repeating elements (it is better structured if the same elements such as overview, discourse and summary are found again and again, alternating with the various topics); a rhythm of question and answer in lessons; presentation slides that separate topics and provide orientation are both boundaries and alternations to the content slides; the headlines in research papers are alternating with text. In programming, a loop with various parameters or a recursive call is an alternating repetition.

2.6.3.5 Positive space

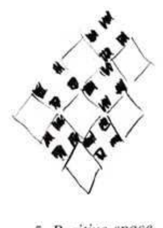


Figure 25. Positive space

"Positive Space is the way that a given center must draw its strength, in part, from the strength of other centers immediately adjacent to it in space" (Alexander, 2002a).

Positive space is, confusingly enough, what is usually called "negative" space by designers, i.e. the background in a figure and ground relation. Alexander points out that this space between the entities or centres forms itself a centre that has to be in harmony with its surrounding. The space between two buildings is itself a shape and it has to be balanced as well. Leitner (2007) sharpens this picture and speaks of a positive complimentary, such as popular sport events are complemented by an audience that is exited about the sport. Coplien (1998a) considers it as “the space waiting to become a center; a latent center”, in education we know open spaces as a conference format, or find positive space in time for individual work or informal setups.

Gabriel (2008b) writes: “a center that moves outward from itself, seemingly oozing life rather collapsing on itself”. This means that a positive space does not negatively infer with the objects themselves, for example wonderful flowers loose their magic if they are pressed tightly at the stalls of a supermarket. A flower needs to be surrounded by space to support it.

2.6.3.6 Good shape

6. Good shape

Figure 26. Good shape

“Good shape is the way that the strength of a given center depends on its actual shape, and the way this effect requires that even the shape, its boundary, and the space around it are made up of strong centers” (Alexander, 2002a).

It seems that good shape is rather a complex quality of interacting centres that together form a good shape, i.e. a strong form or a clear gestalt. It is similar to the law of Prägnanz in gestalt psychology (Quillien, 2008). There are different causes for Prägnanz. Something can be clear because it is the simplest interpretation of the perceived data; another cause is the familiarity with a form - known forms or categories can more intuitively be grasped.

Good shape has an aesthetical meaning. It means that something is the way it should be. Good means to be a good exemplar of a category, an ideal in a Platonic view or a good schema in an Aristotle view. Gabriel (2008b) calls it “a center that is beautiful in itself” so one can say that good shape means to be in a state of harmony and balance – to be beautiful.

Coplien considers good shapes as an abstraction (Coplien & Appleton, 1997) or as simple shapes “that are pleasing the eye (triangles, circles, stars, etc.)” (Coplien, 1998a). This view is coherent with the law of Prägnanz. However, complex shapes that are familiar can also become simple: the familiarity makes a complex entity a simple entity.

For software design, Wirfs-Brock (2010) considers good shape as “a shape that comprises recursive compact coherent centers, each exhibiting characteristic properties [such as] roles and patterns of collaboration, layers, sub-assemblies, modules, domains”. Waguespack (2010, p. 17) interprets Good Shape as overall correctness: “Good Shape brings us to the point of examining the core of the concept, the essence of choices themselves. Together, the collective of choices constitutes the knowledge and understanding of the system under consideration. Relevant, complete, clear, and concise are the characteristics of choice quality, its Good Shape.”

In pedagogy a good method is one that satisfies its purpose and goals without violating the feelings of the students or teacher. A good HANDS ON PRACTICE is one in which the student actually practices and has a positive and sustainable learning effect, e.g. the application of a math skills under supervision of a teacher or trainer. A bad shape of HANDS ON PRACTICE is one that is too hard (students drop out), too simple (students get bored), without guidance (students getting no feedback or help), or does not practice the skills that should be gained. Good shape means both, a method or pattern that fits to the context (the intent) and its right implementation.

2.6.3.7 Local symmetries

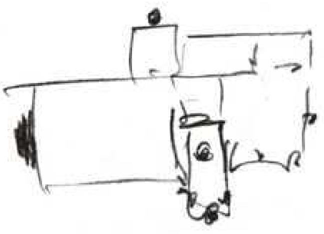


Figure 27. Local symmetries

"Local Symmetries is the way that the intensity of a given center is increased by the extent to which other smaller centers which it contains are themselves arranged in locally symmetrical groups" (Alexander , 2002a).

Local symmetries are the result of symmetry-breaking, the process by which patterns are shaped: "Symmetry breaking is a phenomenon that distributes global symmetries locally" (Coplien, 2001). The random noise or snow on a TV screen has no symmetry at all; a global symmetry on the other hand is given on a blank paper where each area is perfectly symmetric to all others. Both states obviously lack any interesting centres. A local symmetry provides order locally and contrast to other elements. "Local" implies that there is symmetry at one space but asymmetry (broken symmetry) in relation to another space. Symmetry is related to the gestalt law of similarity which states that similar objects are grouped. Forming a group implies that there are objects who belong to the group (centre around it) and other objects that do not belong to the group. Between the members of the group there is a similarity (or symmetry) whereas other objects are in contrast (asymmetrical).

Symmetry implies proper ratio, balance and equilibration. It is well understood that symmetry is often perceived as beauty (mathematically and psychologically) (Steward, 2007). In order to create a shaped form the symmetry needs to be broken, i.e. be local not global (Ball, 2009). Gabriel (2008b) calls local symmetry "a center with another nearby which is somehow an echo" which gives a hint that echoes are a variant of symmetry.

For software design, Wirfs-Brock (2010) refers to "symmetrical behaviors, intention-revealing names, common language and consistent naming, methods containing same level of code detail". Those are all good examples for symmetries that exist between local entities. Consistent naming does not mean that everything should have the same name; but the style of language should be the same. Symmetrical sizes of methods at one code level make the code clean and tidied, maintainable and alive.

Leitner (2007) considers the relation between teacher and students as a local symmetry. However, most of the times there is no symmetry between teacher and students because they differ in power, knowledge and skills. A symmetric relation is rather given between two students who are working together. The asymmetry between teacher and student is not a negative thing. If the teacher would be symmetrical to the students s/he would be one of them without the special role of providing feedback, help and guidance. A global symmetry in the classroom would be bad. Local symmetries such as cooperation, student-centred activities or shared spirit and interests between teacher and students are more valuable than global symmetry. A global lack of symmetry can be a disaster as well: it is the chaos in the classroom where every student just makes what he wants, ignoring any common learning goals or the instructions of the teacher.

2.6.3.8 Deep interlock and ambiguity

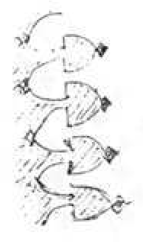


Figure 28. Deep interlock and ambiguity

" Deep interlock and ambiguity is the way in which the intensity of a given center can be increased when it is attached to nearby strong centers, through a third set of strong centers that ambiguously belong to both" (Alexander, 2002a).

This property is about centres that cannot exist without other centres. One centre makes up the other. They strongly depend on each other. A teacher becomes a teacher in the classroom only because of the activities that are performed in the classroom; at the same time the activities in the classroom come into being only because of the role of the teacher. Learning environments, students, teachers, activities, education technology, and resources all derive their role and meanings because of the other stakeholders. Another facet is that each person or object can play many roles at the same time since "group members never belong to only one group. Instead, each person is a member of many groups in different social contexts" (Schümmer & Lukosch, 2007). Likewise, a class in a modularized system can play several roles at the same time. The various roles an object can play depend on the contexts it relates to. The meaning of one centre depends on the context in the same way as the meaning of a word depends on its context of utterance. The meaning comes from the word and the context as well. We find "centers that are hard to pull apart; centers that derive power from surrounding centers; centers that cannot be removed without diminishment; centers that are part of several others" (Gabriel, 2008b).

2.6.3.9 Contrast

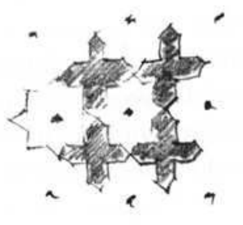


Figure 29. Contrast

"Contrast is the way that a center is strengthened by the sharpness of the distinction between its character and the character of surrounding centers" (Alexander, 2002a).

In general, contrast means that properties or forms have perceivable differences, such as the contrast between colour (black – white, red – blue), volume (silence - noise), size (small – big), social properties (poor – rich), writing styles (narrative – poems), forms of communication (one to many – one to one), attention (following a teacher – make your own thoughts) etc. Contrast means a differentiation in space. This differentiation can be found in geometrical space as well as in semantic space (see section 3.2.1.2).

Two objects can be distinguished because they are different. One figure comes into being because it has perceivable properties that are different to its ground. From the law of similarity which says that similar things are grouped it follows that dissimilar things are separated, i.e. create a distinguished centre. Contrast means broken symmetry.

Common contrasts in education are different activities, methods and forms: individual, small-group and whole class teaching differ in their style of communication; working with whiteboards is different to reading a book; writing is different to listening. The identity of a class, a university, or a group is also a contrast. If the contrast is not implicit, it may be created: the outline of a letter separates figure and ground; different colours of soccer uniforms separate players on the field.

Examples from the world of software design include different method names, different classes, or different coding styles. We can identify a variable by its name; the contrast between two variable names is stronger than the contrast between two numeric addresses in the memory of a computer – for human beings the variable names "price" and "soldItems" have a stronger contrast than memory addresses such as #232532 and #255252. For Waguespack (2010) contrast means identity and clarity of distinctiveness.

2.6.3.10 Gradients

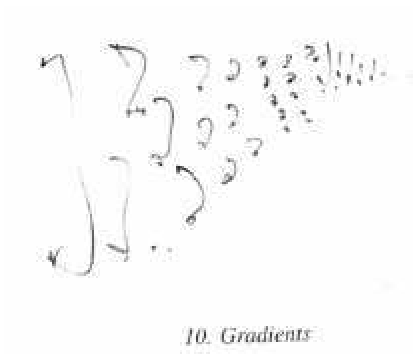


Figure 30. Gradients

"Gradients is the way in which a center is strengthened by a graded series of different-sized centers which then "point" to the new center and intensify its field effect" (Alexander, 2002a).

A gradient is a centre that differs from its surrounding centres by a simple variation - a mild contrast – but keeps most of the similarities. The variation gradually increases into a direction, i.e. some properties change step by step. Instead of an abrupt change (a strong break of symmetry that separates two entities), a gradual change connects and distinguished the centres. The change is softly, "qualities vary subtly, gradually slowly" (Gabriel, 2008b). Between two centres there is a local symmetry; however the local symmetry within one centre is even stronger.

Gradual variations in pedagogy include increasing levels of difficulty for the same type of activity, different grades in school, or grades of student performance (Forschungswerkstatt, 2009). The maturity and experience of a person gradually changes, gradually taking more responsibility or building degrees of trust (Schümmer & Lukosch, 2007) are also examples of gradients. This is what we mean if we say that somebody growths with his tasks or has to grow into it. Abrupt changes can be harmful: a sudden success can make a person arrogant or insecure in the new situation; if there is a huge difference in competences, skills or salary between two levels in an organization there is always a source for conflict.

2.6.3.11 Roughness

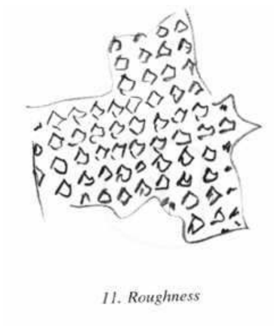


Figure 31. Roughness

"Roughness is the way that the field effect of a given center draws its strength, necessarily, from irregularities in the sizes, shapes, and arrangements of other nearby centers" (Alexander, 2002a).

Roughness seems to account for the special needs of a local situation. It seems to be the property that one finds if parts are locally adapted rather than using pre-made modules that are plugged together like Lego bricks (Quillien, 2008). In this sense, roughness means the individuality of a particularized general form. While an abstracted pattern captures the invariant of a form category, roughness means the variations in particular instances. A category that has only members without variation (such as components from mass production) implies that there is no adaption according to the particular needs. Such forms are not alive because they cannot appropriately interact with their environment; there is no freedom to play with a form and unfold it as the space demands (no "Spielraum").

Roughness means the application of a pattern in the million ways without ever doing it the same way twice. Gabriel's (2008b) hint that "the inessential is left messy" seems to be important: the members of one category have an invariant character in the essential parts but they also have variations according to the local needs. Wirfs-Brock (2010) interprets roughness as code or an implementation that "exhibits some aspect of hand-crafting to fit into its environment". Instead of using pre-made software components, code needs to be unfolded to the specific human, organizational and technical requirements. Agile adaption is imperative to fit software to end-user needs and not the other way around. Waguespack (2010) also derives user friendliness from the potential of local adaptions.

Likewise, ready made lessons or activities that cannot be adapted to the needs of a class will fail; a micro script of social activities that does not allow variations to the needs of a situation often fails (Kollar, 2006); a presentation that has been memorized word by word is boring, the pigheaded application of rules can lead to undesired outcomes. There is no need to re-invent the wheel or established forms of education such as a SEMINAR. But the form needs to be adapted for each single design, i.e. a SEMINAR needs to be adapted according to the content, the skills of the students, organizational conditions etc.

2.6.3.12 Echoes

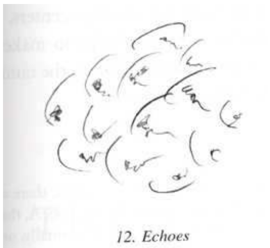


Figure 32. Echoes

"Echos is the way that the strength of a given center depends on similarities of angle and orientation and systems of centers forming characteristic angles thus forming larger centers, among the centers it contains" (Alexander , 2002a).

Echoes are structural similarities between centres, i.e. recurrent centres or structures. They can be interpreted as variations of the same pattern. Leitner (2007), Gabriel (2008b), and Coplien (1998a) all follow Alexander in characterizing the property as a kind of family resemblance. Family resemblance refers to the structural similarity between objects of one category. A good example is a table. One cannot enumerate a number of features that are similar for all tables. Two tables may not have much in common but they can be linked to each other by a chain of objects, i.e. table A and B might be very different, but both are similar to table C.

Leitner (2007) lists some illuminative examples: dogs have similar body postures and behaviours allowing unknown races to be recognized as dogs; specific forms of roofs and building materials create family resemblance between buildings in the alps; the similar structures of hotels, medical centres, or company head quarters.

In educational settings we find recurrent forms of lessons, methods, room configurations, repetitions of content, redundant presentation of information. An interesting relation to levels of scales is highlighted by Schümmer & Lukosch (2007): "Echos point to the fact that we can find the same patterns of social interaction at various levels of scale. Problems like membership and interest arise equally at a community level with thousands of potential members as at a small group level where two to ten members might interact."

In software design, echoes can be found in object instantiation or specialization (Coplien & Appleton, 1997), the handling of errors, ways of decomposing responsibilities, and more general in relationships including complex and basic structures (Wirfs-Brock, 2010).

2.6.3.13 The Void

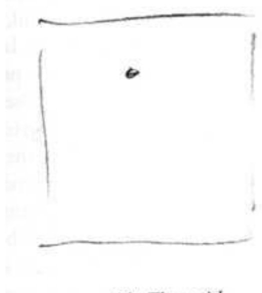

13. The void

Figure 33. The void

"The void is he way that the intensity of every center depends on the existence of a still place – an empty center – somewhere in its field" (Alexander, 2002a).

It seems that the void is a place of calm where centres can relax. There is not an intensity of centres in the void – it is the opposite (or contrast) to centres. As we have seen before things come into being by their contrasts. Maybe one interpretation of the void is that the existence of centres needs spaces where no strong centres are. A lot of centres, an overwhelming of structural differentiation, are like noise. In the void, however, we can clearly see even a small centre. Coplien (Coplien & Appleton, 1997) comments: "'The Void' results from periodic 'cleaning out' or 'self-organizing'. Centers, or groups of centers, will sometimes become too abundant and/or too intense. The resulting conglomeration begins to look a little chaotic or 'too busy' (perhaps even confusing). 'The Void' is a kind of protective response to preserve the structure of the system by purging some of the overly-intense centers, replacing them with a 'homogeneous emptiness' which differentiates and clarifies some of the smaller less visible centers (perhaps ever heretofore unnoticed)."

One could say the void is a place where things can happen because of the calm instead of noise. It is "stillness or literally a quiet point" (Gabriel, 2008b). In pedagogy, the void can mean methods for concentration, to let one's mind wander, time to relax, calm down in the classroom, sinking into a book.

2.6.3.14 Simplicity and inner calm

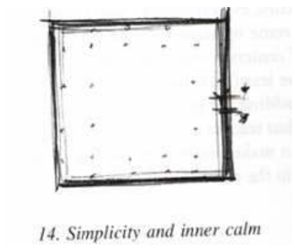


Figure 34. Simplicity and inner calm

"Simplicity and inner calm is the way the strength of a center depends on its simplicity – on the process of reducing the number of different centers which exist in it, while increasing the strength of these centers to make them weigh more" (Alexander, 2002a).

Simplicity means to have an essential form which is not more complicated than it has to be. If we have two algorithms, processes, or activities that mean essentially the same, the simpler one is better because the other one increases complexity without need. Here more complexity means more complicated because unnecessary complexity is noise. In software design, simplicity means "refactored, clean code; sparing use of programming language frills; lack of excessive features in a framework" (Wirfs-Brock, 2010). In science, the principle of simplicity is found in "Occam's razor" (Quillien, 2008): a theory that makes fewer assumptions than another is supposed to be better.

Things that are not essential to the form are noise and can disturb the thing itself (Schümmer & Lukosch, 2007): "Simplicity and inner calm argues for group processes that attain the group's goal without much distraction.". In education, text books should use simple, illustrative examples without distractive decorations. Activities should focus on the subject and not draw away attention (Forschungswerkstatt, 2009). Schematic drawings or models reduce concepts to the core, making them easier to grasp and to understand what is important. All irrelevant parts are taken out. Any further reduction would harm the meaning.

2.6.3.15 Not-separateness

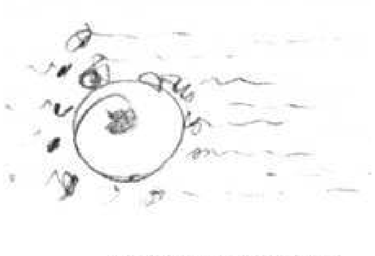


Figure 35. Not-separateness

"Not-separateness is the way the life and strength is merged smoothly – sometimes even indistinguishable – with the centers that form its surroundings" (Alexander, 2002a).

This is another form of deep interlock and ambiguity; each centre is at the same time a self-contained unity and yet depends on other centres. Once again the context plays an important role to make one centre whole. The meaning of a centre unfolds from its own structure and its environment. A separated centre means that it does not smoothly integrate with the environment – the form does not fit to its context. If a centre is not separated, it belongs to its place, it is "at one with the world, and not separate from it" (Gabriel, 2008b). The term "centre" implies not-separateness because there need to be relations that link to the centre (in opposition to isolated patterns or parts).

Pedagogical examples include the curricular units that are linked to each other and each unit influences other units. The courses should fit to each other. The way lessons are designed depends on their disciplines. Schools and universities are not isolated organizations but embedded in a society that puts expectation on them. "Not-separateness reminds group members that the group is part of a larger social system and that exchange with this system is important for the group process" (Schümmer & Lukosch, 2007).

2.6.4 A discussion of the 15 fundamental properties

The 15 properties have been discussed on several of the last PLoP conferences. However, only a small minority of the pattern community is aware of Alexander's later work and the 15 properties. Among those who are familiar with the properties there is still no agreement what to do about them: should we accept the properties and adapt our design decisions accordingly? Or are they rather esoteric principles? The discussion about the 15 properties can be very inspiring but one has to appreciate that there still is a lot of speculation about their meaning. The comments and interpretations given in the previous sections are not the final word on the matter. We can identify some challenges to the properties. The obvious one is that there exists already an abundance of interpretations which gives the concepts a degree of arbitrariness. Alexander claims that the properties improve a design by making it more alive. But are the properties necessary or sufficient? This raises the question whether one can really pin down wholeness to only 15 properties. There is also much redundancy between the properties and one can ask whether there are more fundamental concepts behind the idea.

2.6.4.1 An abundance of different interpretations

The 15 properties are on a high level of abstraction and do not directly translate to all domains. Thus, there are many different interpretations of their meaning. Waguespack (2010, p. 12) maps the 15 properties of centers onto choice properties for system design: "To apply Alexander's concepts of physical structure to information systems, they must first be translated from a language of physical space to a language of cognitive space where physical positions and distance correspond to concepts and consonance in 'fields' populated by abstractions rather than shapes. The term choice serves well for that translation of Alexander's term center into this cognitive space". Some of his mappings are intuitively comprehended, for example the mapping from Boundaries to Encapsulatoion or Good Shape to Correctness. Other mappings are less comprehensible and are in conflict with other interpretations, for example the Void to Programmability or Positive Space to Modularization. Likewise, there are incosistencies in the interpretations given by other authors.

2.6.4.2 Necessity and sufficience of the 15 properties for vivid designs

For each of the properties we had identified examples that showed how the very property improves a design or the lack of the property indicates bad designs. Alexander provides in his volumes *The Nature of Order* similar examples for landscapes, buildings, animals and nature in general. But what does it means that we can find examples that show a correlation of the properties with good shapes? Are the properties necessary, sufficient or both in order to create good forms?

If they are necessary, this would imply that we cannot find a single good design without some of the properties because the properties are necessary for goodness. If they are sufficient then all designs that possess the properties would be great, in other words, we cannot find a bad design that possesses the properties. Even if the properties truly correlate with good design they do not explain how to exactly apply them. In particular, they do not explain functional qualities of forms in the world. Where patterns elaborate about the specific reasons for forms in the "forces" sections and justify the design decisions, the 15 properties are very general. Patterns are stable centres and many of them do have configurations in which the 15 properties emerge. The wisdom captured in patterns, however, is more specific about the configurations.

2.6.4.3 Completeness of the 15 properties

One can also question the number of 15 properties. Alexander points out that he is not sure about the exact number, but the magnitude seems to be right. However, there are more than 100 gestalt laws, and a collection of 100 universal design principles can be found in *Universal Principles of Design* (Lidwell, Holden & Butler 2003). One of the principles in that work is "affordance". If "simplicity" and "the void" are fundamental properties should not "affordance" also be one? The book *Meta-Patterns* (Volk, 1995) provides an analysis of recurrent general forms or properties in nature, including spheres, sheets and tubes, borders, binaries, centres, layers, calendars, arrows, breaks, and cycles. Those meta-patterns are sometimes close to Alexander's properties but many are different yet convincing as well.

While we can easily find more properties that emerge in good designs, we can also make an attempt to reduce the number of qualities because there are many overlaps. Maybe each property can be derived from a combination of purer properties. We can also try to hierarchically relate the properties according to their similarities and differences.

2.6.4.4 Relations between the 15 properties

Waguespack (2010) has clustered the properties based on their coherence. The "coherence is calculated between two properties as the sum of the overlap of their supporting properties – their interrelationship or influence on one another" (Waguepack, 2010, p. 48). The interrelations of properties are noted in a table provided in *The Nature of Order* (Alexander, 2002, p. 238). Unfortunately the resulting cluster hierarchy is rather arbitrary. It seems that the qualitative assessment of related properties cannot simply be used for a cluster analysis that assumes quantitative coherence between the properties. For example, *Echoes* and *Simplicity & Inner Calm* end up in one cluster; another cluster contains *Local Symmetries* and *The Void*. Some clusters actually contain properties that are rather similar, such as the cluster of *Strong Centers* and *Boundaries* or the cluster of *Gradients* and *Deep Interlock & Ambiguity*. Other clusters, however, contain properties that are counterparts, for example *Good Shape* and *Roughness*, or the cluster of *Contrast* and *Not-Separeteness*. Hence, some clusters include properties of similar principle, other clusters have properties that balance each other, and some clusters are rather arbitrary.

Most of the 15 properties have both similarity and contrast. This differentiates and connects them at the same time. For example, *Similarity* is found in *Levels of Scale*, *Boundaries*, *Alternating Repetition*, *Local Symmetry*, *Deep Interlock* and *Ambiguity*, *Gradients*, *Roughness* and *Echoes*. *Contrast* is found in *Level of Scale*, *Boundaries*, *Alternating Repetition*, *Local Symmetry*, *Deep Interlock and Ambiguity*, *Contrast*, *Gradients*, *Roughness*, *Echoes*. Likewise, self-containment and not-separateness are principles in most of the properties: *Strong Centers*, *Boundaries*, *Positive Space*, *Deep Interlock and Ambiguity*, *Gradients*, *The Void*, *Not Separateness*. Each of the properties makes it possible to identify the centre itself while it depends on the environment – the connection between context and form. Connected centres form themselves centres – as in patterns of patterns – and the coherence of the whole is expressed in properties such as: *Good Shape*, *Simplicity and Inner Calm*, *Strong Centres*.

2.6.4.5 Relevance to this work

After discussing the 15 fundamental properties we have to ask whether they can help us to find better patterns or at least get a clearer understanding of the qualities of good patterns. Similarity, recurrence, contrast and variation certainly play an important role for the differentiation of space (3.2.1.1) and the abstraction of similar yet different parts (3.2.7). Since each pattern is a complex entity of several features, two centres will be similar in respect to some dimensions, and different in respect to other dimensions. Similarity connects several centres to one larger whole (not-separateness, smooth variations in gradients, repetition in echoes). Contrast allows for local adaption (roughness) and is what actually shapes the forms – without contrast everything would be dead (e.g. the world would be like a huge blank sheet of paper). Symmetry and symmetry breaking (3.2.10) lead to the emergence of patterns. This is reflected in many of the properties: "Symmetry breaking also plays an important role in the characterization of Alexander's 15 structural properties. Local Symmetry requires some symmetry to be broken. Roughness, Gradients, Echoes, and Level of Scales all exhibit symmetry, but lack perfect symmetry characteristic of the next largest symmetry groups (perfect smoothness, bilateral symmetry, and equal size)" (Coplien & Zhao, 2001). Similarity and contrast discriminate abstract categories and concrete entities in the world. Two features are either similar (similarity) or different (contrast). Contrast "creates" the forms (symmetry breaking), similarity connects the forms (the parts of one form, the members of a category). The human brain does not only perceive separate objects based on similarity and contrast, as gestalt theory suggest, but also memorizes schemas based on similar cases and differentiates schemas based on contrasting cases (3.3.2). Centres that are strengthened by the 15 properties might be more harmonically in the human perception because they balance similarity and contrast, repetition and adoption. Fitting centres help people to organize the world in meaningful ways. As such the 15 properties might play a role in the construction of strong and coherent schemas (3.3.5). However, the schemas of a person depend on individual experiences (3.3.7). While the properties can increase the coherence of a perceived situation, individual expectations and prior knowledge will affect the judgement as well. Thus, we can express doubts that the 15 properties alone account for the "degree of life" in a given context. But if the properties indeed help to find self-contained wholes that are not separated from their environments they could support the process of pattern mining (3.4.4). A pattern that is strong should show many of the 15 properties because it generalizes over similar cases (echoes), can be distinguished from other patterns (contrast), connects to other patterns (non-separateness) on different levels of granularity (levels of scale), can be adapted to specific needs (roughness), can be interwoven with other patterns (deep interlock and ambiguity) and has an overal coherence (good shape).

3. Theoretical framework

3.1 Introduction

This chapter will provide three different views on patterns in order to better understand the nature and the meaning of patterns. We will first discuss patterns in the real world as recurrent structures. Recurrence implies that we assume the existence of universals and consider patterns as things that really exist. But particular instances of patterns are always different in some aspects. Therefore, we need to discuss the different ways of abstraction and the implications on the complexity and information content of a pattern.

That patterns are real does not mean that we all share the same patterns. People have different views of the world and the way of looking at the world shapes what we can see. How people organize the world into meaningful patterns is a question of psychology. We will explore the relations to schema theory to understand the patterns in our minds.

The patterns in the heads of experts are shared in pattern descriptions. The goal of the pattern community is to share the expertise about good practices. However, an explication of a pattern structure in the form of a pattern description is not the pattern itself. Rather it is a network of assumptions about this pattern – a theory about a good practice or design. Therefore, we will consider implications from epistemology for the pattern approach.

In the three main sections of this chapter we will discuss these different views on patterns: the patterns in the world, the patterns in our heads, and the documented patterns. We will start with a short overview before we discuss the concepts in detail. In the conclusions of the thesis we will refine our theoretical framework and consider the interaction between the various patterns and their "places".

3.1.1 Real world patterns

The pattern approach very often has the viewpoint of realism, i.e. it is assumed that the patterns (recurrence of structure) really exist in the world. The problem, however, is not only to find the patterns but also to find an appropriate levels of abstraction and granularity. Even for a simple seminar we cannot capture all dispositions. The reason is that we can extend the list of features of a seminar in many ways: the events during the seminar, the preparation of participants, attitudes of participants, history how the seminar evolved, alternatives in the curriculum etc. This makes it hard to decide what belongs to a proper description of a seminar or other educational forms.

If we pragmatically accept that there is no objective way to describe a whole structure we still face the problem that we have to decide which sub-structures are meaningful patterns rather than arbitrary patterns. It does not make sense to describe half of a wheel or half of a seminar despite the fact that both the half wheel and the half seminar are recurrent parts of the world. We are interested in whole forms. Which granularity is appropriate to divide the world into different patterns is quite a challenge.

At which level of abstraction patterns should rest and where to draw the boundary to other patterns cannot be decided formally and objectively. Rather it depends on the conscious or unconscious decision of the beholder. Moreover, it is an act of personal judgment whether a certain recurrent design actually is a good or best known practice.

3.1.2 Patterns in our heads

Even if there is a structure of the world that is real and objective, the order we impose on the world depends on each individual (Watzlawick, 1976). The perception which parts of a structure are considered as a unit depends on subjective judgment. That is not to say that two individuals cannot put the same or similar order on the world, making the units inter-subjective. Nor is anything said whether the mechanisms that let us order the world are different or the same for each individual. However, since each individual has different experiences s/he will order the world in her or his own ways.

In this process we construct our world as constructivism suggests. The ideas what belongs to a good lecture, a proper seminar, a beautiful online training, or a successful Wiki differ between individuals. One psychological theory that explains how our pictures of the world are constructed is offered by schema theory. It is assumed that our knowledge about the world is stored in cognitive schemas and the construction of a mental model is due to the activation of such schemas (Schnotz, 1994).

If we find a design that is already known to us in terms of its structure, this new design will be assimilated into an existing schema. If we find a new learning system in the web, the implicit structures and properties of that system will let us integrate it into an existing schema we already have, e.g. our conception of a blog or Wiki. In this process the cognitive structure is further strengthened, our idea of what a Wiki can do and which features most or all Wikis have, is getting clearer. At the same time we will find differences to our existing concepts in every new system we discover. A new Wiki system might challenge our previous understanding of what Wikis are because we might find new features or an unfamiliar visual interface of highlighting WikiWords. This will cause an accommodation of the schema, i.e. the existing mental structures are changed in order to adapt to the new experience. Assimilation and accommodation will eventually lead to an equilibrium of the knowledge structures. That is we get a fairly stable understanding of a concept (Piaget & Inhelder, 1969).

3.1.3 Explicit pattern descriptions as theories about patterns

Patterns are documented to share proven solutions with other peers. They do not only describe the forms in the world but also capture some of the personal knowledge about the forms, such as its qualities, the reason for the form, and ways of creating and using it. Patterns are usually described in a special format that accounts for the fact that a form is a solution to a problem in a certain context. Hence, patterns are not only recurrent forms in the world but a text genre to document practical knowledge as well (Gabriel, 2002). As we have seen, patterns are not direct projections of the world but rather projections of individual – or intra-individual if patterns are mined in teams – knowledge structures. The explicit knowledge representations in pattern documents are projections of projections of real structures accordingly (see figure 36)

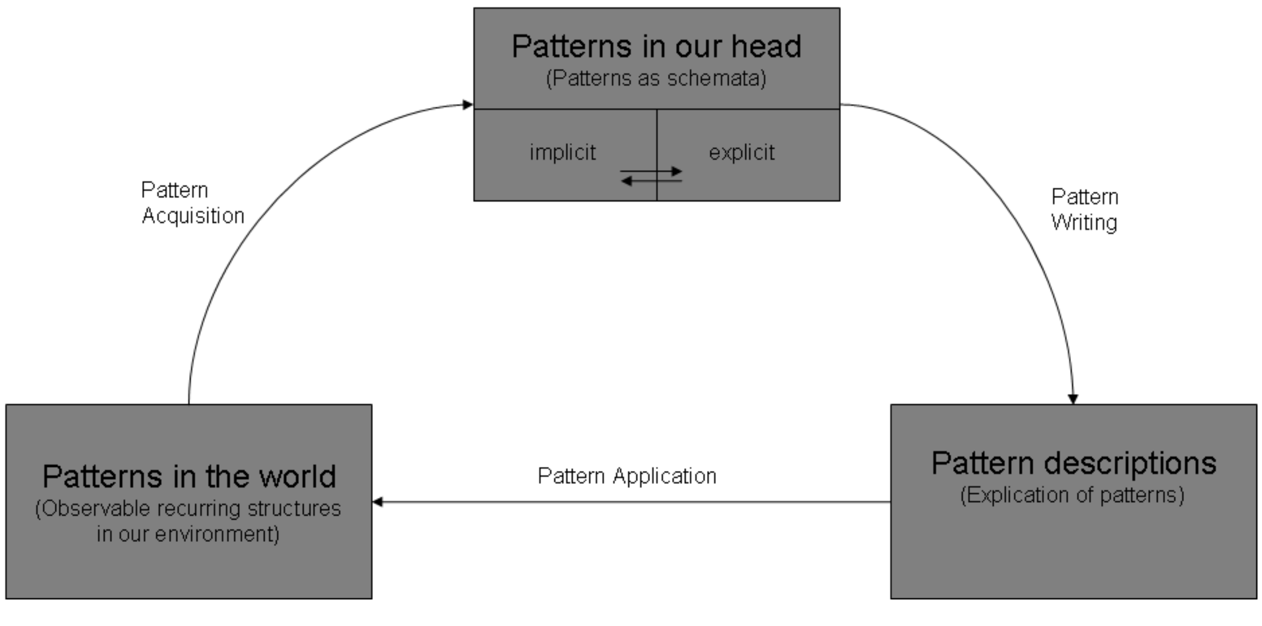


Figure 36. The relation of patterns in the world, patterns in the heads of designers and pattern documents

Once patterns have been identified and mined, a typical description format is used and the patterns are named. However, the text format is just a tool for mediating the patterns. It does not guarantee that what is written as a pattern actually is a good and useful pattern that actually exists in the world. Patterns are derived from practical experience and not deduced from theories. Discovering a pattern is called pattern mining. This metaphor emphasizes the analysis of existing design structures and the implicit knowledge of experts. The process of pattern mining reveals “nuggets of wisdoms” from the structure and form of artefacts and the decision making of their creators. To expose the invariant structure and discriminate it from the surface structure (non-essential features) is the main task of pattern mining.

This mining is an attempt to find the regularities and generative rules of design forms. In the third section of this chapter, we will argue that patterns are specific kinds of theories and that the process of pattern mining is similar to scientific discovery. Exploring the concepts of induction, deduction and abduction with respect to patterns, we reflect upon common methods of pattern mining in the pattern community. This allows for a critical discussion of the level of confidence and corroboration of patterns.

For the scientific scholar the section offers arguments that pattern mining is a research process with outcomes as reliable and sound as other scientific procedures. This justification is needed to establish the pattern approach as a scientific methodology beyond the scope of the pattern community. At present, it is unusual to present scientific findings in the pattern format. Usually, patterns are just illustrative examples in research publications. We will argue that patterns are theories as well and cannot be separated from theoretical findings.

For the pragmatic pattern practitioner**,** e.g. users and authors of patterns, the section encourages the critical reflection on the pattern concept. Patterns are not tried-and-true per se. Similar to theories they have to be subjected to empirical tests. Discussing the validity of patterns is not just belle étage philosophy. Understanding the epistemological nature of patterns is crucial to derive criteria for pattern quality, e.g. the degrees of corroboration, and the limits of objectivism – especially since patterns are not only descriptive documentation but normative instructions, designed to have an impact on shaping our environments.

3.2 Real world patterns

We are one but we are not the same.

Lyrics from One, U2

In this section we will consider patterns as realized forms that recur. It will be argued that a form differentiates space and that it is structured by ordered and related parts. We will first discuss what space means and how its partitioning into parts is a realization of forms. For illustrative reasons we will create a small artificial world structure. While we can see the underlying structure of that world in its totality we will find that even for that fictive structure there are different ways of realizing forms and patterns, i.e. alternatives to derive different realities. However, some realizations are more efficient, coherent, adequate or even true to the underlying structure. Sections 2.3.4 and 2.3.5 discuss these effects rather informally by examples of extracting forms, patterns and hierarchies of patterns. The examples illustrate how the complexity and amount of available information changes by choosing adequate sub-structures or parts of the world. The implications will be linked to information and complexity theory (section 3.2.6 and 3.2.8) and the effects of abstraction (section 3.2.7). Since our chosen views, mappings and abstractions on the artificial world structure alter its reality we will reflect upon the ontological status of patterns by discussing qualities and universals (section 3.2.9). Instead of seeing universals only as generalizations we will argue that a universal is shared by an abundance of structures that belong to the same form category. The differentiation of one universal into a multiplicity of actual patterns will be discussed in a section about symmetry and symmetry breaking (3.2.10). A universal, then, is not only in the end-products but also in the rules and constraints that generate forms of the same quality. Generators (3.2.11) are a formal concept to describe the generation of valid structures rather than specific example structures. The discussion of real world patterns will end with implications for design patterns.

3.2.1 Space

Alexander constantly refers to patterns as the differentiation of space. The particularization of a general pattern is a differentiation of space, too. Though Alexander acknowledges that patterns always correlate with patterns of activities or processes, his diagrams focus on the differentiation or unfolding of geometrical space: towns, buildings and construction. In other domains, such as software design, education or organizational structures, the space is not geometrical but includes other semantic dimensions. Since patterns always happen in space, we will start with a discussion of what perspectives on space exist.

There are two basic views on the organization of space. One considers space as an absolute continuum with territorial roots. Space is an expansion of matter and a location is that place in space (or that part of space) taken by a body (Dünne & Günzel, 2006). Descartes points out that the term “space” is consequently used for an expansion in height, width and depth (Descartes, 1644). The Cartesian coordinate system is a way to organize space into distinct parts. We can place and locate objects in such a coordinate system.

The other view taken on space is a relational one. Instead of considering expansions, the relations between elements are what constitute space. This topological view was promoted by Leibniz who considers the order of elements as space. In such a view an absolute space is no longer in charge as a reference system (Dünne & Günzel, 2006).

3.2.1.1 Differentiations

What is common to both views is that space depends on differentiation. The relation between two elements implies that we have two distinct elements. In an absolute reference system, the space taken by one element is different to the space taken from another element. A Cartesian coordinate arbitrarily differentiates the space continuum into similar parts in order to explicitly express laws and invariants in space. The resolution of the coordinate system defines how differentiated we can refer to the world: If we split a line into 1000 linear segments, the positions 560 and 561 are different. If we split the same line into 100 linear segments, the positions would be considered the same because everything between 56 and 57 would be considered as the same location. This fragmentation of space allows us to refer to specific locations both in space and time (if we consider time as another dimension).

More general, space is ordered by the repetition of boundaries, limits or literally the difference between two segments (Dünne & Günzel, 2006). The original meaning of the word "measure" was "limit" or "boundary" (Bohm, 1981, p. 150). A relational view would consider this difference as relations between elements which are – relative to each other – at distinct locations. Hence, the location is not set by an absolute reference system but by its relations to neighbouring elements (Focault, 1967). Each location is distinct to all other locations and it is defined by its ratio to other locations as well as relations of proximity, direct neighbourhood, to be over, under or in between, etc. (Bourdieu, 1994).

3.2.1.2 Semantic space

Such relations and differentiations are not only possible for the physical space and time. The nature of a point in space is that it can be differentiated to other points. Likewise, we can differentiate along other dimensions. Educational dimensions could be group size, lesson duration, volume of curriculum etc. There are also non-linear dimensions that cannot be expressed metrical, for example characteristics of students and teachers, or room equipment. The discrimination between objects is always based on concepts or categories in our mind (Löbner, 2002, p. 20). Perceiving an object means to draw a line between it and the rest of the world (Hofstaedter, 1980). Such nominal dimensions are divided into different segments where each segment is homogeneous (more or less – depending of the level of differentiation). In software design, the differentiation of space is achieved by coding. Gabriel points out that the "geometry" (in reference to Alexander's strong focussing on geometrical structure) lies in the code itself (Gabriel, 1996, p. 68).

3.2.1.3 Data

In general, we can say that the differentiation along geometric or semantic dimensions creates space. A specific location in space can be called a datum if it can be distinguished from others. Floridi (2010) defines a datum as "x being distinct from y, where x and y are two uninterpreted variables and the relation of 'being distinct', as well as the domain, are left open to further interpretation". This "further interpretation" means that it is somewhat arbitrary – as with all coordinate systems – where we draw the line between to different segments. For example, in the statement "the group size is eight", group size is the variable and eight is a datum that is clearly distinct to group sizes of seven or nine. Another interpretation of the variable group size could distinguish between small, medium or large groups. One is a metric space (linear growth of group size), the other is a semantic space consisting of nominal data.

3.2.2 Forms

The term form is a translation of Plato's word "eídos" which means "idea" or "ideal". Plato's view that the manifest objects of our world are just pale copies of an ideal form (Ur-Form) will be discussed later in the section about universals (section 3.2.9). For now, it is only important to see that a form is not the thing itself but a view on that form (the "idea" or "gestalt"). This view is always an abstraction of the real structure – if we look at an apple we cannot see its molecules, its taste, its vitamins, etc.

3.2.2.1 Forms in space

The division of space into parts is what constitutes forms[14]. Space, then, is where forms can happen. Forms are parts of space (Cassirer, 1931). If space is made of relations, then forms are specific configurations in space. Von Baeyer (2004, p. 22) points out eight synonyms for the word "form" that have been given by the writer Paul Young: arrangement, configuration, order, organization, pattern, shape, structure, and relationship. We can say a form always is a pattern, an organization of parts into a structure of related parts.

Structure – ordered and related parts – is what makes a form. This is coherent with the view that forms are parts of space because space itself is a structure, i.e. a relation of elements. A form is a subset of such relations. For example, in a plane all neighbouring points are related. A triangle is only a sub-set of these points. This sub-set (or form of a triangle) can be defined by three points in the geometric space of the plane. Each of the three points is a datum that defines the form. Again, data is not limited to geometrical space: the form of a Brainstorming is defined by its relations of activities, or the structure of a Brainstorming.

3.2.2.2 Forms are structures

The relation of parts is exactly what gestalt theory means with centeredness (see section 2.6.1). A form or gestalt is a unit of related elements – a structure. Centeredness implies that there are many inner relations between the elements of a form but that there are also relations to the environment of the form, as we have discussed in the previous chapter. In a triangle, each point is closer related (i.e. less different in location) to some other points within the triangle then to any point outside the triangle. Only the points at the boundary of the triangle are at the same time closely related to the inside and outside. That is so because a form is always part of a larger form, e.g. a triangle is part of a plane, or a wheel can be part of a car. The structure of a form, hence, always is a sub-structure of a larger structure.

3.2.3 Arbitrary forms

For the following considerations, we can create our own universe of space by artificially setting up a structure. This structure is a graph made of nodes and relations. We will call this space our world structure. Although it is a small world it looks quite complicated because we have trouble to grasp the order in it – if there is any.

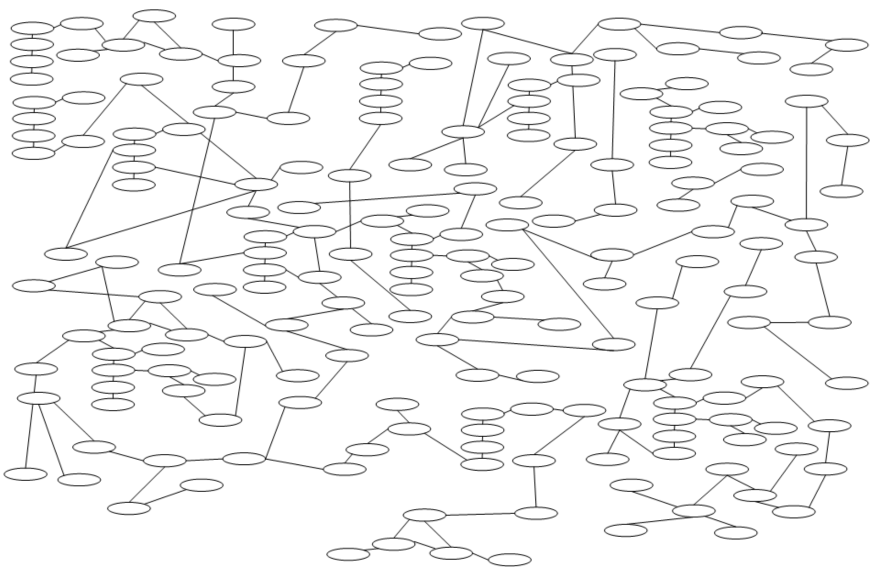


Figure 37. A fictional world structure.

According to system theory, "any given system can be further sub-divided into subsystems. Objects belonging to one subsystem may well be considered as part of the environment of another subsystem" (Hall & Fagen, 1956, p. 20). A simple and naive strategy to reduce the complexity is to arbitrarily divide the structure into sub-structures - arbitrary forms. There are many different possibilities to part that space, some examples are given on the next page.

[14] We can consider this partition as a process of judgement: The German word for judgement is Urteil – literally an original part. An Ur-Teil is a decision whether one part belongs to one category or another. Categories differentiate the world into parts.

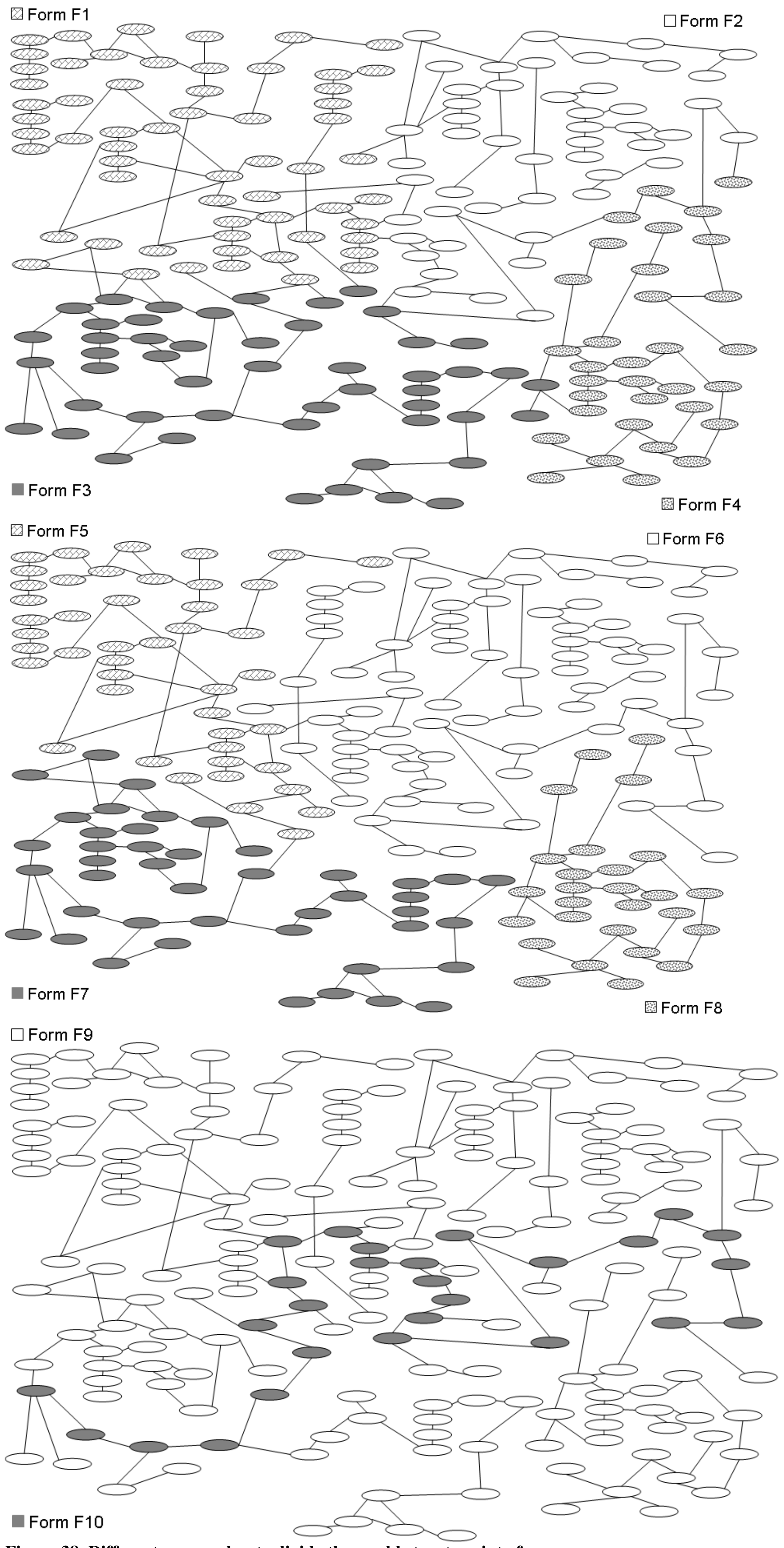


Figure 38. Different approaches to divide the world structure into forms.

We can consider the forms without considering the whole structure. To account for the form's context we include the bonds to its environment as well.

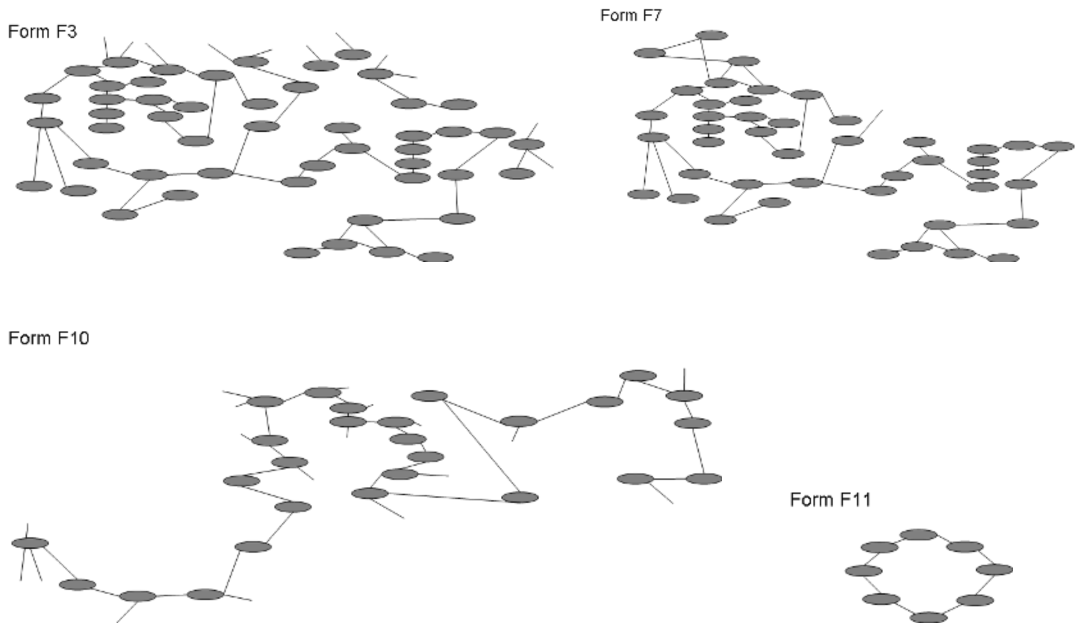


Figure 39. Comparison of different forms in the fictional world structure.

While we can arbitrarily create different forms, not every form is coherent with our world structure. For example, the punctuation of form F11 has no correspondence in our world structure. It is a false view on our world. This illustrates that there can be many alternative ways of looking at a world but not all views are correct.

Each of the other sub-structures is valid but most are not very clever or efficient. For example, form F3 evidently does not reduce the complexity very much. First, it still has many bonds to other forms that we need to consider to fully understand F3. Moreover, some elements of F3 are not even connected to other elements of F3. We cannot speak of a proper encapsulation (see section 2.4.6).

The forms F1, F2, F3, and F4 all cover parts of the overall space, trying to divide the complexity into four sub-parts. Indeed, we can give a simpler representation of our world structure in terms of the four forms F1-F2-F3-F4. But there are still many relations between the parts that need to be understood when considering one form.

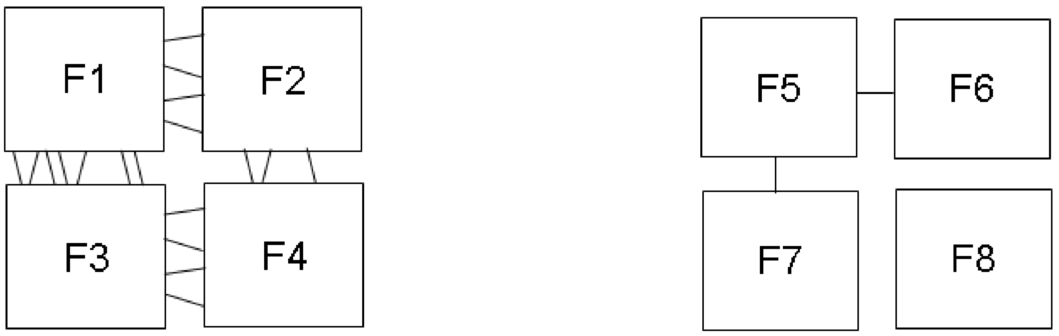


Figure 40. Simple vs. complex relations between sub-forms.

The alternative partition of F5-F6-F7-F8 is an attempt to find four forms to minimize the relations between the forms, in order to consider each part nearly independently. Then we could concentrate on each form without having the other forms in mind all the time. This would account for the criteria of centeredness proposed by gestalt theory and also discussed in *Notes on the Synthesis of Form* (Alexander, 1964) – see section 2.4.1.

Such an order seems to reduce the complexity because we can focus on smaller parts one at a time. The structure composed of the forms F5, F6, F7, and F8 is actually less complex – or at least better to handle. Yet the total amount of structure that we have to understand and store has not changed.

3.2.4 Recurrent forms – patterns

Another attempt to reduce the complexity could be to find recurrent forms. In our world structure, we can identify the pattern P1 which is a form that recurs 4 times in the structure. The benefit of extracting such a pattern is that we have to understand it only one time. This effectively reduces the complexity of the total structure.

Another way of saying this is to note that the world has become less random but more ordered. Whenever we encounter P1 we know its order; we have gained some knowledge about that world. The redundancy of P1 also means that there is less complexity in our world than it may appear on first glance.

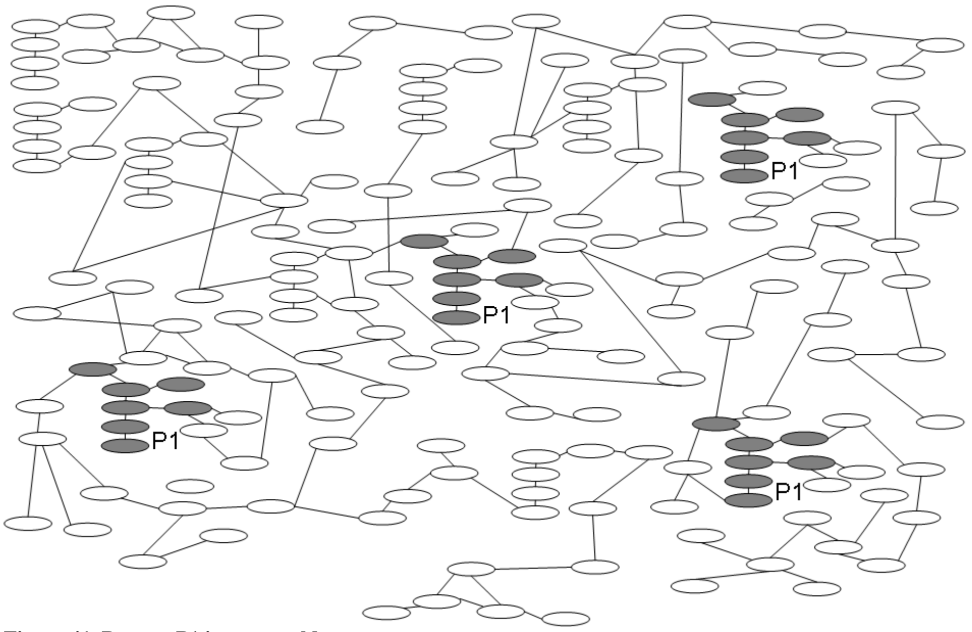


Figure 41. Pattern P1 in our world structure.

3.2.4.1 Amount of information or complexity

To get an idea how the amount of information changes, let us informally consider which data we need to describe the world structure. For each of the nodes we need two pieces of data to describe its position – one for its location on the x-axis, the other for the y-axis. For each relation we also need two pieces of data: references to the two nodes that are linked.[15] Since the number of data required to specify a node or a link is always 2, we could simply say that we need for the node's position a complex datum (x , y) and for the link another complex datum (reference to node 1, reference to node 2). That will simplify our further considerations because we can say that for each node we need one datum and for each link we need one datum for its position.

In our world we have 193 nodes and 192 relations; thus we need 385 data to represent it in a Cartesian coordinate system.

What happens if we replace the redundant structure by the pattern P1? First, to store the form of pattern P1 we need 13 data (for 7 nodes and 6 relations). We also need to store the locations of all occurrence of P1: for the four occurrences we need 4 data. So, to store the pattern and its occurrences we need in total 13 + 4 = 17 data. But we can save the four repetitions of sub-structure in the world structure, which saves 13 data items each time: 4 x 13 = 52. In total, we only need 350 data items to represent the world structure which is a total saving of 35 data.

[15] We can write down pieces of data on a sheet of paper or store it in computer memory. The storage size of each datum will depend on our desired precision.

3.2.4.2 Maximal closure of patterns

Was P1 a clever choice of form or could we do better? If we propose another pattern P2 we will find that it occurs as often as P1 and captures more redundant structure. To store P2 we need 19 data items (for 10 nodes and 9 links) and still 4 data for the positions of its occurrence. We spend 23 data but we save 4 x 19 = 76 data. In total we need only 332 data which is a total saving of 53 data. So, P2 is more efficient.

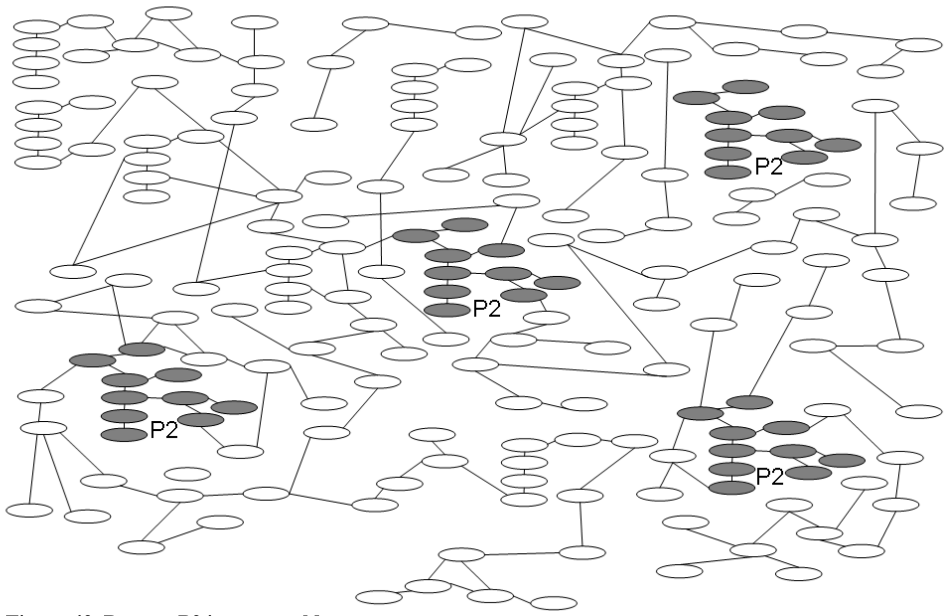


Figure 42. Pattern P2 in our world structure.

The reason that P2 performs better than P1 is that P1 is a part of P2. Hence, P1 does not cover all of the structure that constantly recurs as a whole. Each smaller part of a recurrent structure also recurs: three quarters of a flower occur as often as whole flowers.

If we can enlarge the closure of a sub-structure (i.e. extend the sub-structure by including more related elements) without reducing the number of its recurrence, then we are always better of. The goal is to find the maximum closure.

P2 covers the maximal closure of one recurrent sub-structure because any extension of sub-structure P2 would reduce the number of its occurrences. The maximum closure of a sub-structure captures a whole and self-contained part. Any further extension will lower the number of recurrence: If we extend the form of “a flower” to the form of “flowers in vase” we have extended the form’s structure but we will find fewer occurrences. Yet “flowers in a vase” is a meaningful recurrent pattern (unlike the three quarter of a flower). Therefore, we have to investigate what happens if we find smaller parts as meaningful patterns.

3.2.4.3 Sub-patterns

That P1 is less efficient than P2 does not mean that there are no smaller patterns that are more efficient. Let us consider P3 which is a sub-structure of P2; hence it must recur at least as often as P2. In fact, P3 recurs even more frequently than P2. Do we save more or less data if we choose P3 instead of P2?

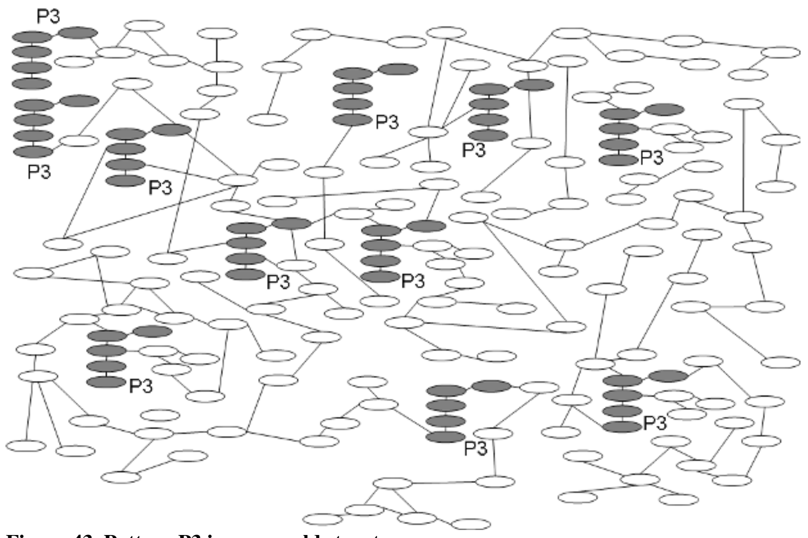


Figure 43. Pattern P3 in our world structure.

The pattern needs 9 data to be stored; for each of the 11 occurrences we save 9 data and need 1 datum for its location. So, we save in total 11 * 8 – 9 = 79 data. As a result we need 306 data items to store the whole world structure. This is more efficient than P2[16].

That a sub-structure recurs more frequently than a larger structure makes intuitively sense: a wheel is more frequently found in the world than a car. Not only has each car multiple wheels but wheels also occur as parts of bicycles, planes or trains.

3.2.4.4 Hierarchies of patterns

If we can only extract a single pattern from our world structure, then P3 is more efficient than P2 in this configuration. However, in our example a single pattern does not account all the redundancy of our world structure. Therefore, we can propose another pattern which covers the remaining structure of P2.

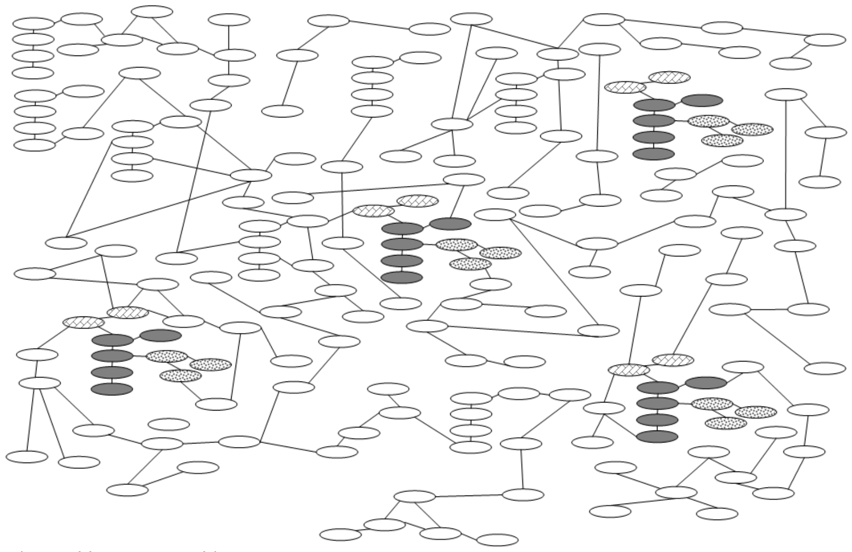


Figure 44. Pattern P4 in our world structure.

The question is should we treat the remaining parts as one pattern P4 or as two patterns P4-1 and P4-2? The answer, again, depends on whether the parts P4-1 and P4-2 occur more frequently than P4. Since this is not the case, it is more beneficial to consider P4 as one pattern because it has a larger closure than P4-1 or P4-2 without lowering its occurrence. To use one larger sub-structure is more beneficial than two smaller sub-patterns because we have to store the location for each occurrence of P4 only once; whereas if we split up P4 into P4-1 and P4-2,

[16] If P3 would occur less frequently – say only 8 times – it would be less efficient (we would only save 64 data in total and need 338 data to store the structure).

we need to store the locations twice. To be precise: in this world, the pattern P4 saves 20 data, the patterns P4-1 and P4-2 save 16 data in total[17].

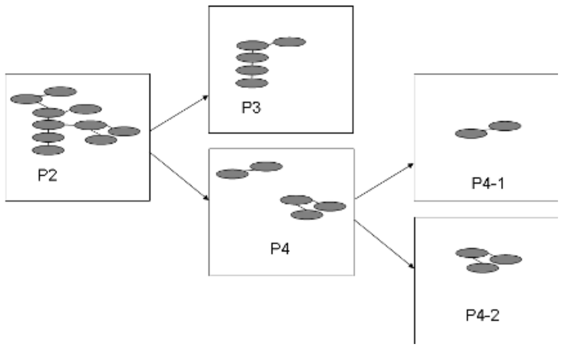


Figure 45. A hierarchy of patterns

But this still does not cover that P3 and P4 and their linkage frequently occur in conjunction. To solve this, we can propose another pattern P5 which has P3 and P4 as related parts.

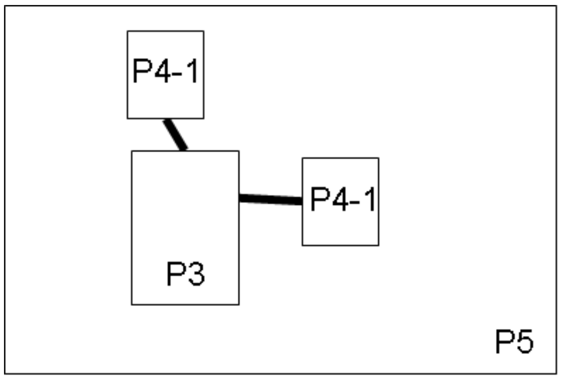


Figure 46. Composition of pattern P5.

This approach has a hierarchical relation where P5 integrates the two sub-patterns P3 and P4.

How efficiently is this approach, i.e. which data do we have to store? In order to define P5, we need to store the relative locations of its two sub-patterns P3, and P4 (2 data), the two bonds that connect P3 and P4 (2 data) as well as an identifier for each sub-pattern (2 data).

The reasons why we need an identifier or reference to the sub-pattern is that we no longer refer to primitive elements such as nodes. We are pointing to a specific sub-structure, and we need to specify which of the patterns we want to use.

So, to store P5 we need a total of 6 data. For each of P5's instance we need another datum to specify its location. In total, we need 10 data to replace all occurrences of P3 and P4 with P5, given that we have stored P3 and P4 already.

To calculate the savings we can compare the effect of P5 and how much data is needed to describe the sub-structures without P5. Without P5 we had to store the locations of P2 and P3 (2 data) each time. Moreover, we had to store the links between P2 and P3 (2 data) each time. Each occurrence cost us 4 data; since there are 4 occurrences we need 16 data without P5.

---

[17] Data size of pattern P4: 5 nodes and 3 links =8 data. 4 occurrences save 32 data minus 4 data for location and 8 data for the pattern. Total savings: 20 data.
Data size of Pattern P-4-1: 2 nodes and 1 link = 3 data. 4 occurrences save 12 data minus 4 data for location and 3 data for pattern. Total savings: 5.
Data size of Pattern P-4-2: 3 nodes and 2 links = 5 data. 4 occurrences save 20 data minus 4 data for location and 5 data for pattern. Total savings: 11.
The total saving of P4 is 20 and the total saving of P4-1 and P4-2 is 16. P4 saves more because the data for the location is only discounted from the savings once.

With P5 we need only 10 data; the total savings of extracting the information that P3 and P4 recur in the same configurations is 6 data.

3.2.4.5 Redundancy in patterns

A pattern that uses other patterns creates a hierarchy of patterns. We could further sub-divide the pattern P3 into two occurrences of sub-pattern P6. We can always find a finest pattern partition for a given structure, i.e. we can identify all “minimal patterns that have no pattern as a proper subset” (Grenander, 1996, p.93). To maximize data compression, no pattern should have unaccounted sub-patterns. If a sub-structure (a pattern) has itself recurrent sub-structures, these are still patterns of a lower level that are not accounted for.

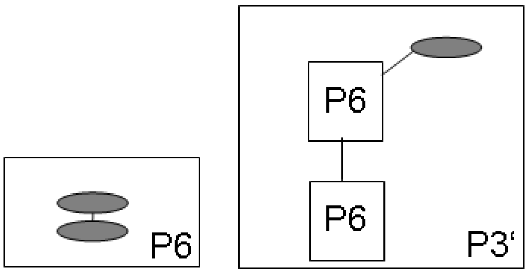


Figure 47. Introduction pattern P6 to cover all redundancies.

3.2.5.6 Overlapping patterns

There are many more recurrent sub-structures in our world structure. Let us consider the pattern P7 which recurs 3 times.

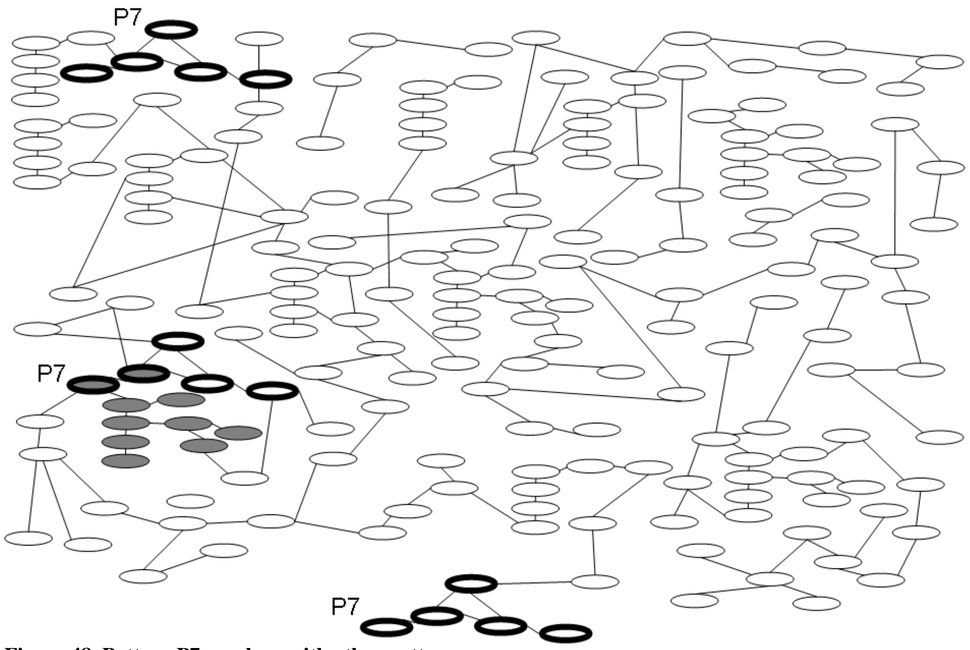


Figure 48. Pattern P7 overlaps with other patterns.

One recurrent sub-structure overlaps with an occurrence of an instance of P5. If we want to replace the sub-structure by reference to a pattern P7 we would over-specify two nodes (because P5 and P7 define the location of the two nodes). That means that we have created a new redundancy.

If some parts of P7 are sometimes also part of P5, should we sub-divide P7 into sub-patterns? None of its sub-patterns recurs more frequently than P7 itself in the world structure. To split P7 into parts does not make sense for the following reasons: While it is true that we have redundantly specified 2 nodes and 1 link (3 data are “over-specified”), to extract the sub-pattern and refer to it from P7 and P5 would cost us more data: the sub-pattern needs to be stored (3 data) and both P5 and P7 have to refer to it and locate it (2 x 2 = 4 data). In total we could save 6 data but would have to spend 7 data.

Moreover, the recurrence of P7 as a whole suggests that the elements of P7 somehow belong together. Such a situation is found if elements of the real world overlap: an object instance in object oriented program structure could belong to several patterns. Activities in education very often overlap: the end of a lecture could be at the same time the beginning for writing down a summary. It is often necessary to let patterns overlap to maintain their individual wholeness.

3.2.4.7 Patterns of different dimensions

There is another way in which pattern can overlap. Let us consider a modified world structure where each node can have a different colour. In this scenario we could still employ the extracted patterns but we have to store for each occurrence of the patterns not only the location but the colour of the nodes as well. One of the instances of P3 could be red, another green; another could have a sequence of different colours for each node. In this case we need to store the colour for each single element of the pattern's instance to reconstruct it.

A close look shows that at least one sequence of colours also recurs more than once in the world structure. We could propose a colour sequence S1 = {yellow, orange, green, purple, red} that stores the sequence of colours; both instances of P5 and of P7 can be combined with S1. In this case the colour pattern S1 modifies the spatial patterns P5 or P7 (see also section 2.4.4.3 about modifiers).

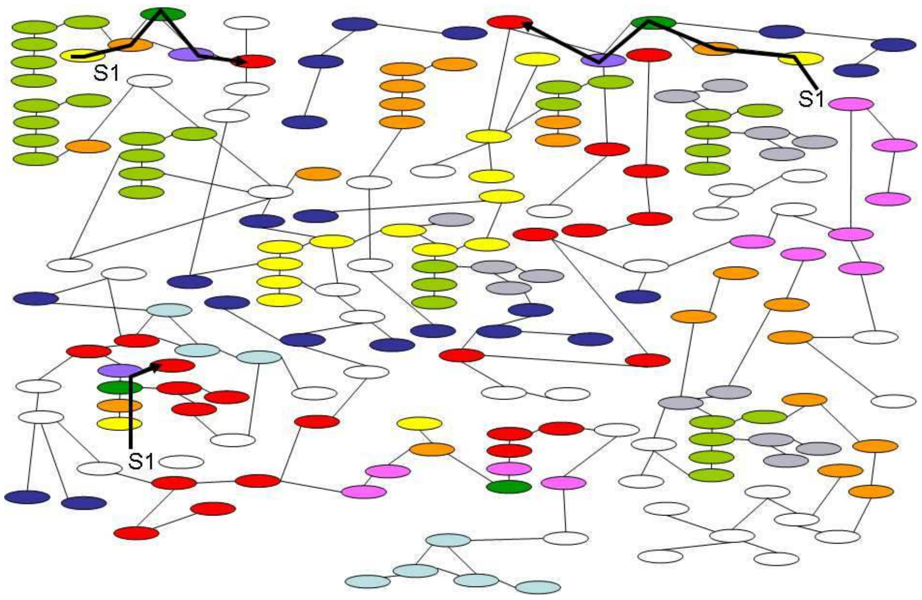


Figure 49. Patterns in different dimensions of space.

In the sequential pattern S1 each succeeding colour is a datum of that pattern. The pattern relates places in a semantic space defined by the different colours instead of different positions on a plane. The variation of colours differentiates that space in the same way as the horizontal and vertical positions differentiate the space of a plane. A pattern of colours such as S1 is an ordered relation that defines colours of nodes. Patterns of positions such as P5 or P7 are ordered relations that define positions of nodes. Patterns of different dimensions can be combined to create multidimensional structures, whereas patterns of the same dimension can be combined into a hierarchical system of patterns.

3.2.5 Information as knowledge

Extracting patterns as we have done in the previous section is a way to reduce the complexity by gaining knowledge about redundancy. Strictly speaking we are not reducing the complexity but we discover that the structure is less complex as it seemed on a first glance. We simplify the structure by pulling out redundant complexity and distribute the remaining complexity among nearly decoupled sub-structures.

3.2.5.1 Ordered elements in structures

The reason why the complexity is lower than the number of elements suggests is that the structure is not all random but has ordered elements. Some specific sub-structures are more probable to be found in the world structure than others.

We can understand the level of order as the information that we have about a closed system (such as our world structure). If all nodes and links were fully random, we could find no patterns or meaningful forms. Randomness means a lack of form or patterns (Von Baeyer, 2004, p. 99). Without patterns we cannot make any statements about a sub-structure without explicitly describing the sub-structure.

3.2.5.2 Informed by patterns

On the other hand, if we have a pattern such as P5 or P7 we can simply refer to the pattern instead of using the actual sub-structure. Of course we have to know the structure of P5 or P7. An uninformed interpreter who does not have the information about the structure of P5 will not understand what P5 means. For example, if we refer to "triangle" we usually do not have to explain or draw what a triangle is because most people know the structure of a triangle – they are informed about the structure of a triangle.

In this sense information can be understood as imposing, detecting or communicating a form (Von Baeyer, 2004, p. 25). By detecting the recurrence of a sub-structure we have detected information about our world; we labelled the information P5 and can use the label to communicate about the structure. If we place P5 somewhere in our world structure, then we are imposing its form at that location.

3.2.5.3 Quantification of information

We had already informally quantified how the amount of information changes by using patterns. We had translated the world structure into pieces of data that specify the nodes' positions and links between nodes. This translation between different notations of the same structural relations is called coding (Gleick, 2011). Another example for such coding is the translation of sound waves into patterns of bits in order to enable communication via computer networks.

The benefit of numerical representation of information is that we can quantify the amount of information that is actually in the system. Consider the following two sequences of data:

1: "abcabcabcabcabcabcabc"
2: "odlrhjdpqwvkmtpalkfgs"

The first sequence of data is ordered. The information contained in it is "repeat 'abc'" or "repeat the first three letters of the alphabet". The second sequence of data is random, i.e. not ordered. It might be counterintuitive but the second sequence contains more information. In the first sequence we always *know* which letter comes next; because we have that information already the next letter in that sequence is not *informative* to us. In the second sequence there is no way to predict the next letter; hence, each next letter is informative to us.

3.2.5.4 Effective communication – Shannon information

So far we have not considered whether the next piece of data (the next letter) is meaningful to us. We have only considered whether the next data tells us something new. This kind of information is also called Shannon information based on a classic paper on information theory by Shannon & Weaver (1949). Shannon was interested in effective communication via information channels (such as a telephone lines). For the effective transmission of data it is only important to consider the structure of the available data (the patterns) without taking meaning into account. In his model a message is sent from a source to a receiver. If the source always tells the same (e.g. repeats "abc") the information content is zero because the receiver knows exactly what is coming – there is no need to send the message because we know the form (the structure of data) already. On the other hand, if the source sends a random sequence of letters, we have no prior knowledge of the next data and the information content is high.

Most messages are neither random nor as regular as a repetition of the same data again and again. The sentence "The TV is broken" contains words and letters that are not randomly chosen. Moreover, the frequency of their occurrence is not evenly distributed: the letter "e" occurs more often than "x", and the word "the" occurs more often than "TV" in human communication.

In this view a less expected letter or word contains more information because we did not see it come. Hence, the probability for each occurrence of data has impact on the amount of information. If we receive a message with less probable data then we get more new information. For example, the message "The sun will shine tomorrow." is less informative than "Tomorrow there will be a tornado" (Taleb, 2007). We might expect that tomorrow there will be sun, shower or heavy rain. The first message can clarify that. But we usually do not expect that there will be a tornado. The message about the tornado is more informative because it is a possible but less probable event. We are more "surprised" by the message.

3.2.5.5 Probabilities of outcomes

A regular sequence of data (such as "abcabcabc") has no element of surprise (Mainzer, 2008); the probability for each next letter is constantly p=1. In a pure random sequence the surprise of each datum is the same; the probability for each message depends on the number of different messages or symbols possible and is p=1/number of different symbols. Most communication systems have different levels of surprise for the various messages or symbols; each symbol has a different probability and the sum of all probabilities is 1. Surprise can also be expressed as degree of uncertainty or data deficit about the next message (Floridi, 2010).

For our world structure we could define an alphabet of symbols where each symbol carries a different sub-structure as a message. The recurrent structures (the patterns) have a higher probability of occurrence. Hence, they provide less information if we find them. We are less surprised to find P5 if we *know* in advance that P5 occurs several times.

The number of different sub-structures and their recurrences is a measure of how much information is in the world structure. If our world structure would be very regular it would not be very informative because we can easily learn everything there is to know about the structure. Once we know its patterns and their frequency of occurrence there is no further information in the structure. On the other hand, if our world structure would be random and disordered it would carry a lot of information itself because it is impossible to use external knowledge (generalized information) about the structure. While there would be more information in the structure it would be less meaningful.

In the previous section we said that Shannon information does abstract from meaning. However, for our world structure we have seen that the alphabet does have a meaning. Each symbol of the alphabet means a specific sub-structure. The meaning of P5 – the actual structure meant by the symbol P5 – needs to be agreed on between a sender and receiver in order to be understood in communication: "While it is perfectly possible to transmit strings of symbols with syntactical accuracy, they would remain meaningless unless sender and receiver had agreed beforehand on their significance. In this sense, all shared information presupposes semantic convention" (Watzlawick, Bavelas & Jackson, 1967, p. 21).

By pulling out meaningful patterns of sub-structure we gain knowledge about the structure. In this sense extracted external information is knowledge. Once we have that knowledge the structure is less informative on the next occurrence of the same structure. We are not getting further information when we encounter P5 another time.

3.2.5.6 Measurements for complexity

A structure that contains less information is also less complex. The complexity does not depend on the sheer number of available data but how ordered these data are related: "the essence of information lies in relationships among bits, not their sheer number" (von Baeyer, 2004, p. 145).

The complexity can be measured in many different ways because there are different types of relations between data. For example, consider the sequence "1, 3, 4, 7, 11,18, …". We can generate an infinite sequence if we know its simple rule. Likewise, pseudo random number generators use simple rules to generate sequences that are random in a statistical sense. If we see such a sequence without knowing the generation rule, we would find no patterns.

This implies that in any structure there can always be hidden regularities that reduce the complexity enormously once we recognize the pattern. Chaitin (1999) and Kolmogorov (see Li & Vitanyi, 1997) proposed to define the algorithmic complexity of a symbol sequence by the length of the shortest program to generate that sequence. Mitchel (2009) summarizes eight different ways of measuring the complexity: as size, as entropy, as algorithmic information content, as logical depth, as thermodynamic depth, as statistical complexity, as fractal dimension, or as degree of hierarchy.

We have already shown that the entropy (number and frequency of different patterns), the algorithmic information content (how to generate nodes, links, and patterns), and the degree of hierarchy (levels of patterns) play an important role to reduce the complex appearance of a structure to its actual complexity. We will further discuss these in the next sections, in particular how complex systems emerge from the interplay of simple rules and which information details are irrelevant in a certain context.

3.2.6 Abstraction and variations

For our world structure we had extracted redundant information and stored it in a way that we were able to exactly reconstruct the world structure. This is a case of lossless compression of data. It is the same principle applied when we compress files on our computer. A zipped file contains the same amount of information as an unzipped file; its file size is smaller because most of the redundancies have been removed. Likewise, a GIF or PNG graphic file compresses the exact structure of an image.

However, there is another way to compress a graphic file. For the human eye small colour differences (e.g., two very similar reds) may not be noticeable. If a number of slightly different colours are treated as one colour, a picture becomes more regular: a sequence of 100 different reds becomes 100 times the same red. However, we can never reproduce the image exactly. Compression with loss means that we loose information about the exact data because we map from variations to an abstraction. For the human eye the exact data of colour may not be significant. Likewise, MP3 compresses audio data because it treats different frequencies not distinguishable for the (average) human ear as the same. But when and why can we afford to loose such information in general?

3.2.6.1 Variations in structure

We had no trouble to identify some patterns in our world structure because some of the sub-structures are exactly the same (except for their locations). However, consider a slightly modified world structure.

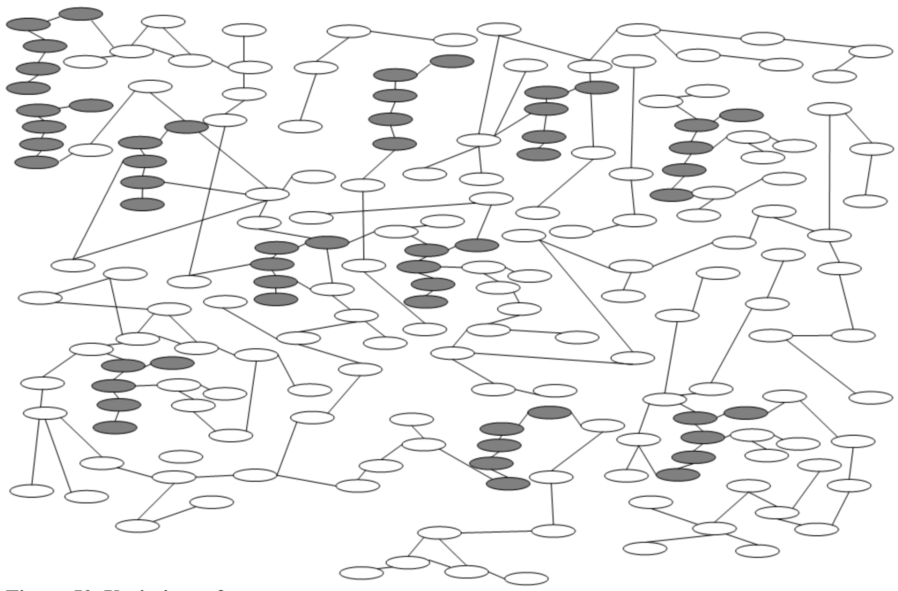


Figure 50. Variations of patterns.

In this structure, the pattern P3 does no longer occur exactly. Rather, there are variations of P3. The sub-structures look similar because they have the same structural relations. The differences are found in the actual positions of the nodes. We can still use the structural relations of pattern P3 but to reconstruct the exact instances of P3 we need to store the exact positions of the nodes. The question is: do the small variations of positions matter or can we treat the different occurrences as equivalent?

3.2.6.2 Abstraction and mapping

Variations can only be seen as identical if we interpret some differentiations equivalently. In mathematics, abstracts usually define equivalence classes. Different abstraction concepts (Kaschek, 2004) or abstraction principles (Fine, 2002) relate objects of a given domain by a criterion of identity. Abstraction also plays an important role in software development: “The first requirement in designing a program is to concentrate on relevant features of the situation, and to ignore factors which are believed irrelevant. Abstraction thus implies simplification. That is, we reduce, at each stage of specification, the amount of information – of concepts and their interrelation – which we must hold or manipulate, when considering that situation” (Bjørner, 2006, p. 234). Depending on what we are interested in different abstraction concepts will be applied, including classification, generalization, and aggregation (Kaschek, 2004), abstraction to properties or mathematical models such as sets, maps or lists (Bjørner, 2006).

Whatever abstraction principle is at work, an abstraction is a mapping from the actual world to a model. Such mapping is a function, f: $X \rightarrow Y$, where the value of X is mapped to another value Y. The coding of our world structure into numerical data is an example of such a mapping. Maps of landscapes are mapping the actual structure of a landscape into a diagram.

Maps in general are models that correspond to some other structure. If $X = \{x_1, x_2, \ldots, x_n\}$is a list of details to be modelled, then $Y=\{ y_1, y_2, \ldots, y_n \}$ could be a list of corresponding features in the map (Holland, 1998, p. 30).

However, as the saying goes, the map is not the territory. A map does not only transform the original structure to corresponding features (e.g. using a different scale) – it also skips many details and alters the phenomena: "[abstraction] does not accept the phenomena as they are, it changes them" (Feyerabend & Terpstra, 1999, p.5). For example, a map may not show the trees, flowers or stones of the actual landscape. Different diameters of a street are reduced to a line of constant thickness. Moreover, maps often use symbols to represent certain types of entities in the landscape: cities are symbolized by a dot or cottages are represented by a small house icon[18].

Leaving out details, considering different ranges of values as equal, and categorizing different objects into one class, are different ways of abstraction that occur in the process of mapping. Each means a loss of information:

Abstraction by isolation. A feature of the actual object is omitted. Positive abstraction means to isolate the relevant features and focus on them; negative abstraction isolates irrelevant variations between several objects (Erdmann, 1892). Features that are obmitted in an abstraction will have no correspondence in the resulting model. For example, if we create a computer model of a car to simulate its driving performance, the colour of the car may not be relevant. In the discussion of our example world structure we have never paid attention to the representation of nodes as ellipses. Hence, we could abstract from that feature. It would do no harm to our world structure if we used rectangles or stars to represent the nodes instead.

Abstraction by generalization: A feature of the actual object is covered but not precisely. While isolating abstraction omits irrelevant features, a generalization abstracts from irrelevant variations of values but preserves the general structure (Wundt, 1907). For example, the different widths of a street are mapped to one line thickness. The variations of a node's position in pattern P3 could be mapped to one exemplary position.

Abstraction to emergent qualities: A feature could emerge from different configurations of structures. For example, a map does not differentiate between the particularities of specific cottages. Likewise, the different occurrences of P3 could be treated as one form category. Abstraction to emergent qualities is discussed in the context of both information and complexity theory (for example Holland, 1998; Mitchel, 2009; Gleick, 2011) but the idea of complex abstractions (i.e. the complex grouping of features to a whole) has already been discussed by St. Thomas (Bobik & Sayre,1963).

An abstraction reduces and maps the available information to the essential information – for example the information needed to follow a path. The art of creating maps or models is to find out which features are essential and which are irrelevant in a given context.

3.2.6.3 Mapping functions

If we want to map the different positions of nodes in the sub-structures of our world to an "idealized" pattern P3, we have to choose a mapping function that maps each actual position x_1 to the corresponding position y_1. We can choose the mapping function arbitrarily though not all functions make sense. These are some proposals:

1. f1: Map the positions of x_n to the average of all x_n positions.
2. f2: Map the positions of x_n to the average of $\min(x_n)$ to $\max(x_n)$
3. f3: Map the positions of x_n to a range of positions that spans from $\min(x_n)$ to $\max(x_n)$
4. f4: Map the positions of x_n to one exemplary x_e

Let us consider how the mapping works for a sub-set of four instances of P3: a, b, c, d.

[18] see also section 2.4.2 about abstraction

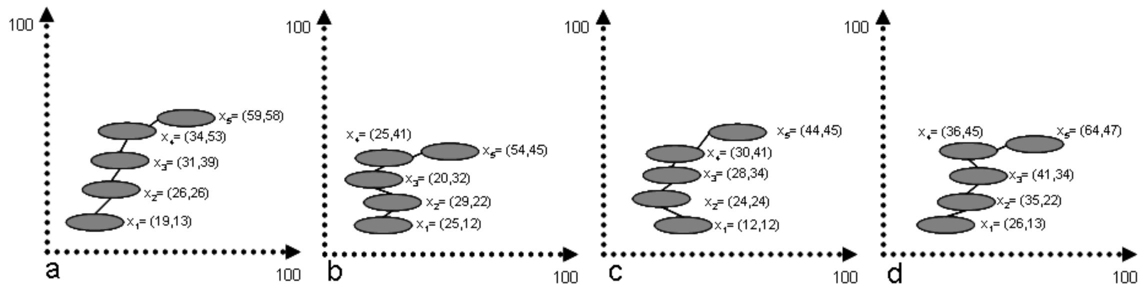


Figure 51. Different instances of P3.

Mapping function f1 calculates for each node the average position. For example, we will take the values of the various nodes x_1 – which are (19, 13), (25, 12), (12, 12), and (26, 13) – and calculate the average. Hence, the mapping to the average of all values would be y_1 = (20.5, 12.5). The average positions of the other nodes would be y_2 = (28.5, 23.5), y_3 = (30, 34,75), y_4 = (38.5, 45), and y_5 = (48, 48.75). None of the mappings matches an actual value of x.

Mapping function f2 uses the average of minimum and maximum values. The minimum value of all x_1 nodes is (12,12) and the maximum is (26,13). Hence; the mapping of the average between min and max would be y_1 = (19, 12.5). For the other nodes it would be y_2 = (29.5, 24), y_3 = (30.5, 35.5), y_4 = (42, 47), and y_5 = (44.5, 51.5). Again, none of the y values matches any value of x.

Mapping function f3 defines for each node the boundary of its possible values:
y_1 = range from (12,12) to (26,13).
y_2 = range from (24,22) to (35,26).
y_3 = range from (20,32) to (41,39).
y_4 = range from (30,41) to (54,54).
y_5 = range from (25,45) to (64,58).

This mapping explicates a range of valid values whereas function f1 and f2 do not capture the acceptable deviation from the average.

Mapping function f4 allows us to use one of the actual instances as a model of P3. For example, we could map all possible values x_1, x_2, …, x_n to the actual values of one of the instances. Furthermore, we could construct a model by using for the nodes x_1, x_2, …, x_n positions from different actual instances: y_1 = x_1 of d = (26, 13), y_2 = x_2 of b = (29,22), y_3 = x_3 of c = (28,34), y_4 = x_4 of c = (30, 41), and y_5 = x_5 of d = (64,47).

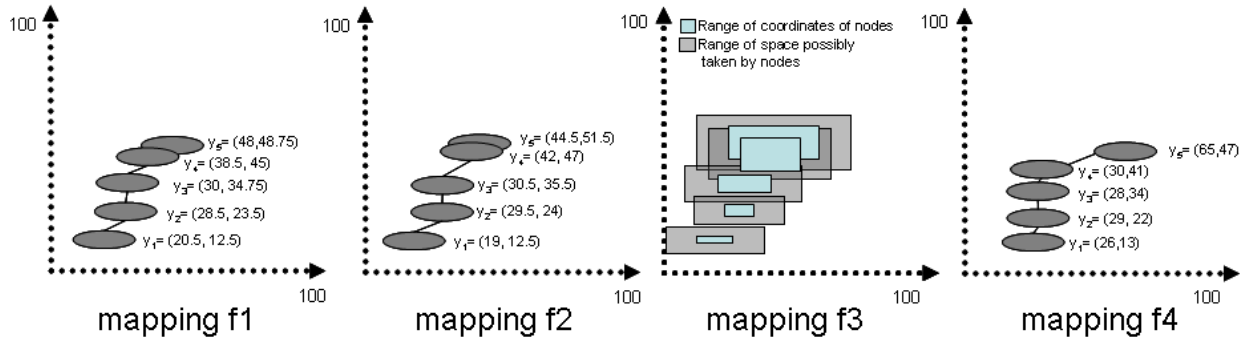


Figure 52. Alternative mapping functions.

3.2.6.4 Models

Mapping f4 is modelled from various instances. It is very similar to the structure of P3 from our original world where all instances were identical. In fact, we could propose a mapping function f5 that maps the values from the instances to that "ideal" P3.

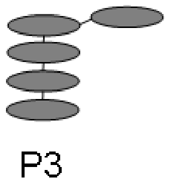


Figure 53. A model of P3.

This structure never occurs exactly in our modified world but it is a good model of P3's structure. It could occur in our modified world in the same way as it occurred in our original world.

What can we learn from these considerations? A model can be a structure that actually exists in the real world, i.e. a model citizen, a model student or a model school. In such cases the model is an instance of the class it models (Goodman, 1976). A model can also be something that we artificially create, e.g. a simulation or a model of molecules to better grasp their structures.

A model can also exist on various levels of abstraction (including concrete manifestations) as long as the essential features of what is modelled are not lost. In a model we find corresponding features for all relevant information. There is an "isomorphism" that preserves the relevant information: "The word 'isomorphism' applies when two complex structures can be mapped onto each other, in such a way that to each part of one structure there is a corresponding part in the other structure, where 'corresponding' means that the two parts play similar roles in their respective structures" (Hofstadter, 1980, p.49). A model preserves the quality that emerges from the complex interaction of the elements of a sub-structure.

3.2.6.5 Implicate order

If we construct instances of P3, we can position each node based on the average values (and their implicit deviations) or within the explicit range of mapping function f3. This allows us to reconstruct the original instances. However, we could also derive structures that have a different gestalt quality than any of the original instances – examples are given in figure 54.

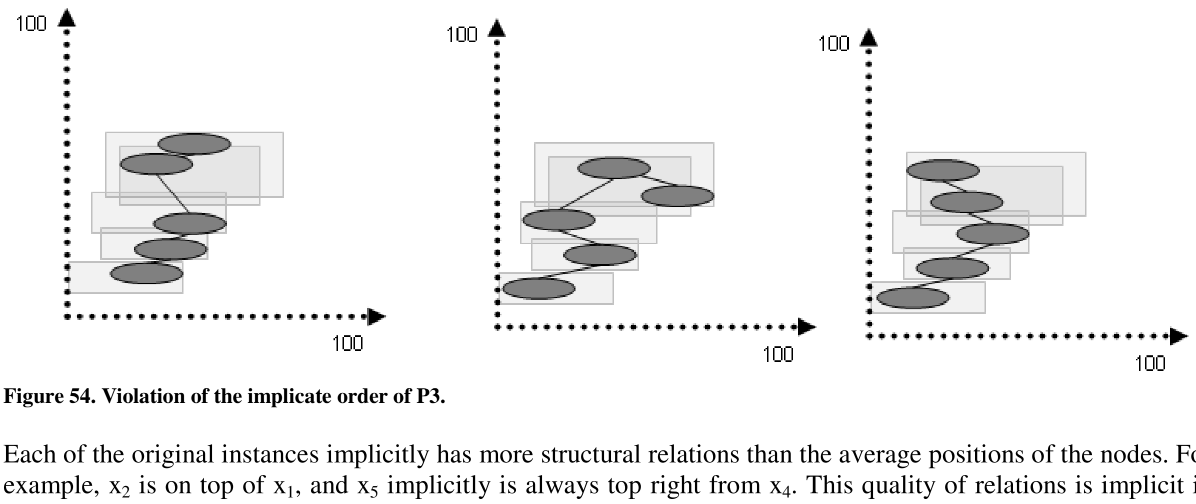


Figure 54. Violation of the implicate order of P3.

Each of the original instances implicitly has more structural relations than the average positions of the nodes. For example, x_2 is on top of x_1, and x_5 implicitly is always top right from x_4. This quality of relations is implicit in the data of each of the instances. It is also found in the model. However, an abstraction that only covers ranges of values does not account that the location of each node depends on the other nodes. This *dependency* is not explicit but implicit in the order of each instance (Bohm, 1981).

3.2.6.6 Emergence

The structural quality of P3 emerges from the interplay of its part not from the single positions of the parts. A global behaviour that outlasts any of its components is a defining characteristic of complex systems (Johnson, 2002, p. 82). The order is not defined by a statistical distribution of each part but the result of self-organizing parts. Elements influence each other and their configurations feedback to the configuration of other elements.

For example, x_5 and x_4 are always in a relation that x_5 is top right from x_4 within certain limits. This feature is implicit in all occurrences. If x_4 moves more to the right, then x_5 has to move more to the right as well. There is also an implicit order between the distances of each node. If we scale the structure, the ratio of the distance between x_1, x_2, and x_3 has to remain constant.

Emergence means that the complex interplay of elements of a lower level leads to new effects or laws on a higher level. The quality of P3 emerges from the interplay of the nodes of its structure. However, we do not need to consider the specific configuration of the nodes as long as the structural quality is preserved: To understand how a car works we do not need to analyse its chemical composition (Von Baeyer, 2004). Different configurations of a lower level (micro-state) are mapped to one emergent quality on a higher level (macro-state). While similar structures, such as P3, are likely to emerge the same quality, there can also be very different structures that emerge the same quality. For example, two very different paths can lead to the same goal or very different situations can make people happy. If we refer to the emergent quality alone we have no information about the actual configuration that produces that quality.

3.2.7 Information as knowledge deficit

The mapping of several micro-states to one macro-state is an abstraction from different qualities on a more specific level to one quality of a higher and more abstract level. Examples are: several positions (micro-states) are mapped to one model position (one macro-state), the level of fuel in a car (many micro-states) is mapped to enough/need to refill (two macro-states), group sizes are mapped to small/medium/large (three macro-states), the forms of cottages (micro-state) are mapped to a symbol (one macro-state), the instances of P3 (micro-states) are mapped to the pattern P3 (one macro-state).

3.2.7.1 Unknown micro-states

Since multiple micro-states can relate to the same macro-state, a reference to the macro-state alone means a knowledge deficit. A reference to the macro-state P3 does not inform us about which actual instance (micro-state) is meant. The knowledge deficit is higher the more variations there are between micro-states. For example a deck of cards consists of 52 cards. If we are informed that a spade was drawn, then we only know that the card represents one of the 13 microstates spade-ace, spade-2, spade-3, …, spade-king. However, learning to have drawn a “queen” is more informative because it reduces the number of possible micro-states to four (queen of spades, hearts, diamonds or clubs). Mitchel (2009) describes a similar example: the different states of a slot-machine can be seen as microstates. The macro-state “you win” has far less microstates than the macro-state “you loose”. If there are multiple ways to reach a design goal, an abstract reference to that goal does not provide any information how to reach the goal. The more alternatives there are the less informative the abstract formulation is.

3.2.7.2 Entropy

The number of microstates that give raise to a macro-state is defined as the entropy of a macro-state by Ludwig Boltzmann (Mainzer, 2008). In statistical mechanics large scale properties (e.g., heat) emerge from microscopic properties. Instead of defining the exact positions, velocity and future behaviour of each molecule in the microscopic level, statistical mechanics uses the average positions and velocities of large ensembles of molecules (Mitchel, 2009). If the different entities of the lower level are structured and ordered in the same or similar way, then we have more information about the actual configuration. However, if there are many different micro-states that give raise to the same property of the higher level, we know less about the actual configuration. That is why entropy measures the “mixedupness” (Floridi, 2010) in systems that bear energy or information. High entropy means that the states of molecules are equally random; low entropy means that the molecules are highly ordered.

The Shannon information in communication measures the message content as the degree of “uncertainty” or data deficit. Likewise, the Boltzmann entropy means a lack of knowledge about the actual structure. Shannon information and entropy are conceptually the same in that they measure the order and randomness of a system (Floridi, 2010).

3.2.7.3 Emergence of qualities and structural similarity

At the level of the macro-state, it does not matter which actual micro-state there is. To know that it is warm in our houses is enough information; we do not need to know the actual configuration of molecules. Although information about a macro-state does not reveal the actual structure of the micro-state it does provide some information about the micro-state. The macro-state "hot" implies different possible micro-states than the macro-state "cold"; the macro-state "iron" implies different possible micro-states than the macro-state "wood".

Likewise the macro-state of the pattern P3 implies different potential structures than the patterns P1 or P2. Though a notion of P3 does not refer to a specific micro-state (knowledge deficit) it does refer only to micro-states sharing a structural similarity. Therefore a reference to P3 informs us very well about an actual universal structure and its related quality.[19]

3.2.8 Qualities and universals

We have discussed several ways to model a sub-structure that ideally represents the several instances of P3. The arbitrariness of choice brings us to a serious question about the ontological status of P3: Does P3 exist or is it a convenient product of the mind? More general we can ask: are patterns real? Since a pattern is different to its instances we are faced with a long dispute in philosophy about the problem of universals. This problem similarly asks whether universal forms exist, i.e. whether they are real. Such questions about the status of reality can be important for modelling information systems and representing knowledge in ontologies (Wohner, 2003).

To understand the problem let us consider two red apples. The question is whether independent and universal properties of "Redness" and "Appleness" exist. Such universals would be abstract objects (Moreland, 2001). That means they are not only epistemological abstractions of concrete objects (e.g. red apples) but "Redness" has an independent existence. More over, the same universal can be found in an object that exemplifies it – the red of the two apples is numerical identical because they exemplify the same "Redness".

3.2.8.1 Platonic ideas

Universals are ideal forms or ideas. Plato believed that the concrete objects are just shadows of the real forms: a concrete triangle never is perfect. No drawn triangle possesses exactly what our idea of a triangle is. Therefore, he promotes a view that the ideal objects have their own existence outside from the concrete, imperfect particular objects - in a place often called the Platonic heaven or sky. However, as Plato points out himself in *Parmenides* (Zehnpfennig, 1997) there are different possibilities where ideas are located: in the objects themselves (objectivism), in the mind of an observer (subjectivism), or beyond the two (absolutism). These are different position of realism, each assuming that qualities or universals are real. As a consequence, we have two worlds: the physical world of individualized objects ("particulars") and the metaphysical world of ideas ("universals").

There are some problems, however, with the assumption of universals. Besides the fact that universals increase the number of existing objects, one could wonder what natural ideal forms there are. If there is an Ur-Horse (the ideal of all horses), is there an Ur-Animal (the ideal of all animals) as well? And what about new forms – is there an Ur-Computer? Moreover, the relationship between an instance and its ideal form is another object which needs an ideal form. This relation again, has an ideal form – leading to an infinite regress.

3.2.8.2 Nominalism

Those objections have led to a counter-position of realism: nominalism or anti-realism. Nomininalists refuse that abstract objects exist. Some of them accept that one can have abstract representation of objects. However, this is an epistemological abstract form created by focussing on some aspects and dismissing irrelevant information; this is different to ontological abstract forms that have their own existence (Moreland, 2001).

Realists believe that there really is a "Redness" whereas nominalists believe that we only abstract to properties that seem to be identical but are not the identical property. Two red apples can look alike but their reds are

[19] In our original world structure P3 even infomed us about everything, i.e. the actual structure. That is the case if a macrostate corresponds exactly to one invariant micro-state.

numerical different. For them, “red” is just a predicate. There is only a class of red things, i.e. a set that contains all instances that are red, but there is no universal “Redness”. There are two main positions of nominalism (Moreland, 2001). Radical nominalists say that an object is “red” if it is denoted by the term “red”. The term “red” denotes a category of objects that are red. Moderate nominalists think that the membership in the class of red objects is due to an epistemological abstraction: the class of red objects consists of objects whose abstract representation is identical. However, they are not numerical identical – red_1 and red_2 in such a set may look alike but they are not the same. The problem of nominalism is that the definition of meaningful classes (such as the class of red things) is hard to justify without indirectly referring to universals. For example, what could justify a class of red things? If it is a similarity between concrete objects or abstract views of them, then this similarity, again, is universal.

3.2.8.3 The reality of pattern P3

Let us illustrate the different positions for our pattern P3. A radical nominalist would say that P3 is the class of P3-objects. This would be a set that contains all occurrences of P3. A moderate nominalist would say that P3 is a class of abstracted objects; if we apply one of the mapping function proposed in the last section, each instance is mapped (transformed or abstracted) to identical forms. Those mapped forms, however, appear identical but are not the same.

A realist would claim that there is a universal quality that is shared by all instances of P3: An objectivist locates the universal in all instances of P3, i.e. we can find the same universal in each of the instances of P3. A subjectivist would say that the idea rests only in our minds; we mentally induce an idea from the various instances. An absolutist would say that there is an ideal Ur-P3 and the instances are just imperfect shadows of that Ur-Form.

So, what is P3? It is unlikely that we will settle a dispute that has such a long history. But we may dare some suggestions and take our own position. The previous sections have argued that the extraction of P3 was possible because we identified information about a recurrent form. The pattern is what is *in* the formation: the implicit order between the elements. If we think about the pattern, our thought idea resembles this implicit order. Each of the instances resembles this order – the instances are symmetrical and isomorph, i.e. one instance can be derived from another by structure preserving transformations. If we map the instances to other notations this implicit order needs to be preserved as well.

3.2.8.4 Where is the pattern?

So, the “ideal” sub-structure that we have proposed for P3 is not the pattern itself – it is just a model that shows the pattern. It exemplifies the pattern because it shares the universal structure of the pattern. This structural similarity is real. However, its reality depends on its instances. Aristotle criticized Plato for assuming that there could be ideal forms beyond the actual instances (Detel, 2005). For him, there could be no universal without an instantiation of the universal. The universal of a horse is in the particulars of all horses. Consequently, if there is another horse in the world, this horse changes the character of the universal.

If that is true, then the universal of all horses that ever lived is different to the universal of horses that live today. It is also different to the universal of all horses that I have ever seen and thought of – my idea of horses is different to your idea of horses. Both of our ideas are different to an objective idea of all horses (which no one could achieve though it is theoretically there).

However, it is likely that the structure of our ideas is symmetrical – there is a universal isomorphism between our different ideas of horses and an objective idea of horses (Hofstaedter, 1980).

We could say that universals are the symmetries between the members of a class. The class of animals consists of all animals as members; the symmetry between animals is their universal. A more specific class such as Birds differentiates its members against other subclasses such as Dogs. Though birds and dogs share the universal of being an animal, dogs do not share the universal symmetry between birds.

3.2.8.5 The problem of abstraction revisited

The problem of abstraction is often that the elements are tackled independently, and an abstraction seeks for what is common in multiple instances. As a result the rich concept of a lecture and its many variations would be reduced to a common denominator. It would be an attempt to find “unity in multiplicity” (Bortoft, 1996). But if we take two instances of a lecture it can very well be that we find no common features at all, for example an online lecture on thermodynamics may have nothing in common with an on-site lecture about pedagogy. They may involve different kinds of rhetoric, media resources, different professors, different durations, different levels of student involvements etc. The remaining common factors (such as “a professor meets his students”) are certainly not characteristic for a lecture nor do they give us any idea into what the concept of a lecture can unfold. Yet we do have an intuitive understanding of what a lecture is, letting us both perceive a lecture and prepare or deliver a lecture. There seems to be something like a universal idea of what a lecture is, and this universal can unfold into the many variations of a lecture. That is to say that the phenomenon of a lecture is not what is common to all lectures but it is the possibility of all lectures given in the past and future. Such a concept is not reductive but constitutive. It is “multiplicity in unity” (i.e., the multiple ways in that a lecture can unfold) rather than “unity in multiplicity” (i.e., the same properties that are found for all lectures) (Bortorft, 1996).

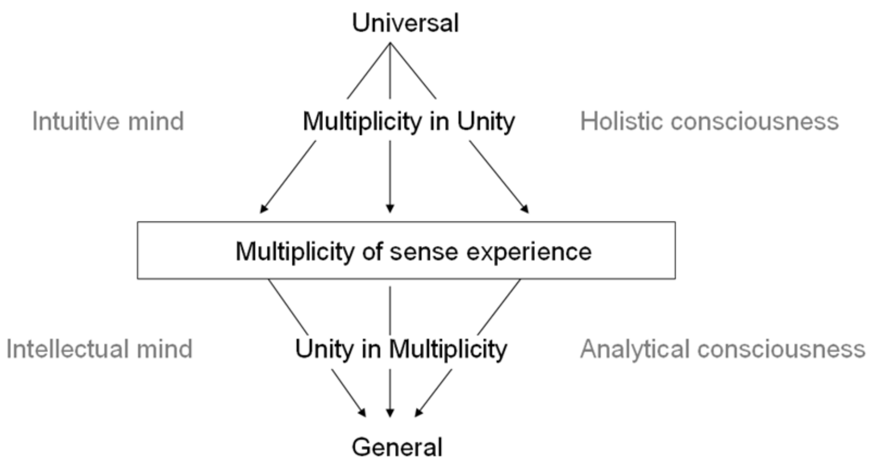


Figure 55. Universals vs. generalizations.
Two different ways of abstraction (Bortoft, 1996). The analytical mind reduces the concept to simple abstraction. The intuitive mind groups related forms into categories.

3.2.8.6 Form categories and family resemblance

Two instances of a lecture are therefore related to each other, but not by abstract commonalities, rather by being unfolded from the same primal phenomenon. “Once the primal phenomenon has been discovered in a single case, it can be recognized elsewhere in nature and in artificial situations where superficially it may appear to be very different” (Bortoft, p. 23). Wittgenstein refers to Goethe’s morphology (see section 2.6.2.3) when he introduces the concept of family resemblance (Buchholz, 2006). The members of a category are not simply related by sharing similar features but linked by a chain of intermediate members with whom they do share features. Two lectures may have no features in common but can be linked by a chain of similar lectures. For example, the “online lecture on thermodynamics” may have nothing in common with the “on-site lecture on pedagogy”. However, it may share features with an “online lecture on pedagogy” and the later may be linked to an “on-site lecture on pedagogy”.

3.2.9 Symmetry and symmetry breaking

The key to identify similarities between parts is symmetry. In a strict sense, symmetry is not a feature but rather a process: a structure preserving transformation (Steward, 2007). A triangle that is rotated is symmetrical because there is a specific transformation that keeps the structure and yet changes the object. Symmetry means change, yet the same (Zhao & Coplien, 2002). What is preserved is the order of the parts. “…symmetry has a duality of change and constancy whereby some aspects of an object can be changed while leaving other key aspects invariant” (Zhao & Coplien, 2003). For instance, all exemplars of an ONLINE LECTURE resemble the same structural relations, yet each concrete delivery of an online lecture is unique. In software design, classes are an example for symmetry, whereas refinements or parameterization break the symmetry (Zhao & Coplien, 2002).

Comparing an ONLINE LECTURE to a different type of an online event such as CHAT we can see that these forms are asymmetrical – there is not a simple symmetry operation that resembles similarities between the two forms. They are only symmetrical in that they can be both used for online communication in e-learning scenarios. But if they are unfolded to their specific levels of an e-learning scenario their symmetry is broken.

In the same way the symmetry of ONLINE LECTURES is broken when some of its parts unfold into different directions, e.g. using different conference systems (with asymmetrical features), having different rhetorical styles, or making use of different collaborative options.

3.2.9.1 Causes of symmetry breaking

The reason for symmetry breaking is that in a particular context or situation, certain forces act on the symmetrical form, forcing it to break its symmetry. "In physics, symmetry breaking happens when a force disturbs a symmetric system and causes it to break its symmetry to maintain its stability" (Zhao, 2008). For example, water molecules moving at random in the atmosphere can coalesce into six-pedalled snowflakes. "The symmetry of a uniform gas can be broken by applying a force that changes the disposition of its molecules" (Ball, 2009, p. 23). If somebody moves along a path in a symmetrical way (e.g. keeping the direction) and encounters a huge obstacle, he is forced to change his direction. The symmetry of his motion is broken. A pattern of motion emerges because a force has caused a change of direction. Symmetry breaking is fundamental to the process of pattern formation (Steward & Golubitsky, 1992). Alexander's ideas on patterns and whole forms are to his own account highly related to the idea of symmetry breaking: "Alexander believes that purely symmetric designs are dysfunctional. This has many repercussions for design. Perfect reuse is perfectly symmetric, so you need parameterization and refinement instead of simple cloning. You can't articulate a design in terms of symmetries" (Coplien, 2001).

3.2.9.2 Unfolding and Symmetry breaking

If we consider the concept of a lecture, it can unfold into on-site or online lectures. An online lecture can further unfold into the use of specific conference systems, specific topics, different lecturers and so on. In each of the unfolding steps symmetry is broken, i.e. the symmetry breaking is making the form of the lecture distinct to others by differentiation.

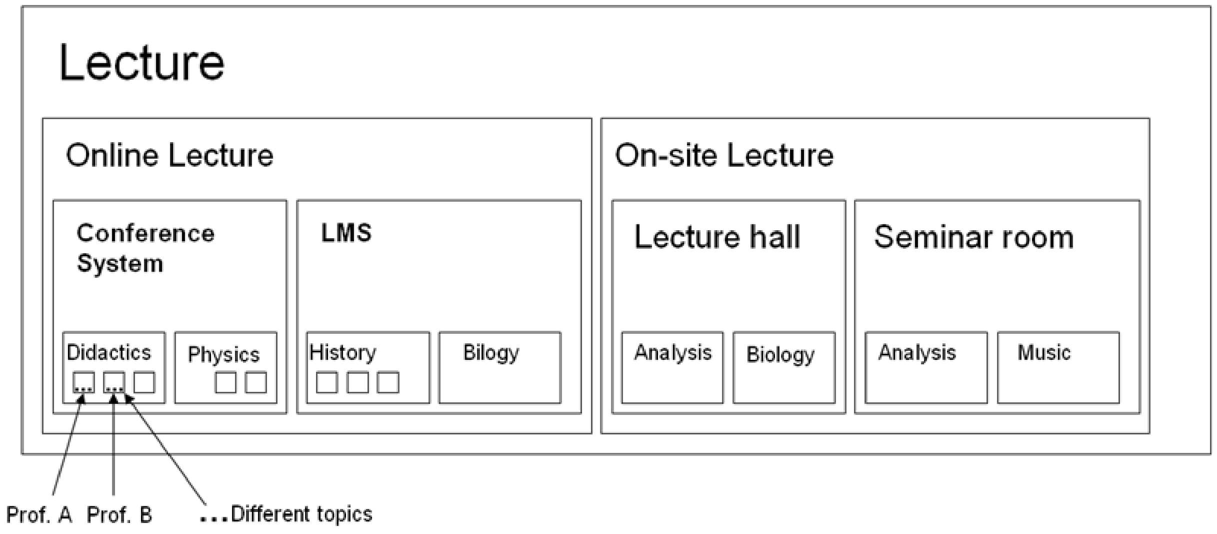


Figure 56. A lecture unfolding into different forms by means of symmetry breaking.

At a first glance, the hierarchical order in figure 57 seems to show simply different levels of abstractions. From an analytical point of view this may well be so, a lecture seems to be more general than an online lecture which is a specific kind of a lecture. But intuitively an online lecture is as much a lecture as any other lecture.

3.2.9.3 Unfolding of lectures

It shows that the symmetry breaking shown in figure 56 is only one example for the sequence of unfolding or differentiation. That the concept of a LECTURE is first divided into ONLINE LECTURE and ON-SITE LECTURE in the beginning is somewhat arbitrary. It could have been that a LECTURE first unfolds into different topics (such as thermodynamics or pedagogy) and then into different delivery forms. However, each unfolding step has consequences for the sequence that follows. Once a lecture unfolds into an ONLINE LECTURE one of the following unfolding steps will be a particular online system. Likewise, a lecture that unfolds to an ON-SITE LECTURE will have the choice of a specific lecture hall as one of the following unfolding steps. This is similar to generating a design by the proper use of a pattern language. Recursively the descision for one pattern implies further patterns of finer granularity (see also section 2.4.8 about the generativity of patterns).

Let us follow the path of symmetry breaking given in figure 56. On breaking the symmetry of lectures the form parts into two categories. Note that a category is not an abstract concept but rather the set of all concrete instances that have broken their symmetry in the same way (e.g. being differentiated into online or on-site lectures). As a consequence, the more often the symmetry is broken (following the levels downwards in figure 56), the overall symmetry between two categories decreases whereas the symmetry within a category increases. The category "lecture" includes all potential lectures; hence two members of the category may be very different. If you consider the categories "Online lecture on thermodynamics by Prof. X – topic 1" and "Online lecture on thermodynamics by Prof. X – topic 2" the exemplars in both categories have broken the symmetry in the same way, hence the structural form that emerges is very similar. The "On-site lecture on pedagogy by Prof Y – topic 1", on the other hand, has broken its symmetry differently many times. Hence, the emerging form is very different. Yet all three lectures emerged from the same universal concept, being part of the same universal category at the beginning.

3.2.9.4 Shaping a form – Gestalthaftigkeit and Gestaltbarkeit

When we communicate patterns the categories of medium levels are of special interest because the members of that category show both similarity and variation. The similarity is due to the fact that the members of that category have taken the same paths in symmetry breaking. The variation is due to the fact that the process of unfolding on that level has not ended and therefore the members still unfold in different ways. It is only at the medium-levels in which we start to grasp a whole form – a gestalt. More and more wholeness is gained in the process of unfolding. The whole is present in each concrete instance, e.g. in each lecture, and therefore in each member of the category "lecture". However, since so many different forms are in that category, an abstract representation of a "lecture" seems impossible. On a medium level the categories contain already members that are more similar to each other. If we create an abstract representation for the members of such a category, we do find similarities that we can embed into our model. The more similarities between the members of a category, the more whole would be an abstract representation because it possesses already much of the actual forms in that category.

More concrete representations capture more of the wholeness ("Gestalthaftigkeit"). A very concrete representation shows a fully unfolded instance that is no longer open to differentiations of design ("Gestaltbarkeit" is missing) (Kohls, 2009b). It is therefore, that patterns are abstractions on a medium-level. They are concrete enough to let us see the whole (the "gestalt") while at the same time they are not finite instantiations. A pattern constrains the forms but it does not fully define them. While there are abstract representations used to communicate and capture patterns, the actual pattern includes all the forms possible. It is a form category and not just an abstract form.

3.2.10 Generators

A pattern – or form category - such as P3 is an infinite space of possible configurations; however, the implicit order of P3 sets constraints to what possesses the quality of P3. For example, there are an infinite number of possible oak trees; yet the quality to be an oak tree sets constraints to a form: a maple tree is not an oak tree. The two categories imply different constraints and regularities. The more general category tree puts fewer constraints and rules on the form.

3.2.10.1 Building blocks

Once we have an idea of what a tree is, e.g. its character or its essence, we can recognize and imagine all possible trees. It is the common part that enables us to build up the "tree" classification (Holland, 1998, p.24). Such classifications are building blocks that we can use to construct our world. In this context, a building block does not mean a ready-made component. Rather, it is a strategy to generate the various instances that fall under its category.

But how can we actually create new instances of an idea or of a pattern? One strategy is to use a model and evolve it in a sequence of structure preserving transformations. For example, we can derive the structure of a new car from an existing car. Likewise, we can derive the various instances of P3 from the modelled example of P3.

3.2.10.2 Building a form

However, exemplary models may not account for all the diversity of instances of a universal. Another approach is to describe the steps to generate a form in terms of simple rules. Each step leads to a specific state of the design. Rules and constraints imply the next step for each state. If we have a small set of rules we can still unfold many different particulars. For example, each of the implicit orders in P3 is a rule: x_2 is on top of x_1, x_5 is top-right of x_4. The relative distances and angles between nodes are implicit rules as well. In order to define valid configurations of P3 we must not consider the single nodes in isolation. Rather, the structural quality of P3 emerges out of the regular and constrained interplay of its components. Likewise, we cannot define a car by only considering its components – we need to understand their interactions.

John Holland compares such emergent qualities of complex systems to games: "The rules constrain the possibilities: not all board configurations are legal, and new configurations follow from legal changes to configurations already achieved" (Holland, 1998, p. 22). Moreover, once we have played a game several times we develop strategies (rules of thumb) how to react in certain situations – a chess player recognizes a pattern and acts accordingly. These rules, or strategies, are similar to the rules of thumbs captured in a pattern language that provides a strategy to solve a design problem.

3.2.10.3 Constrained generating procedures

Holland generalizes this principle by introducing "constrained generating procedures" (gpc). A gpc consists of transition functions that define for each state of a model a succeeding state based on the current state of a model and its input parameters. Since the state of one gpc can be the input for another gpc, we can build more complex gpc by connecting gpcs.

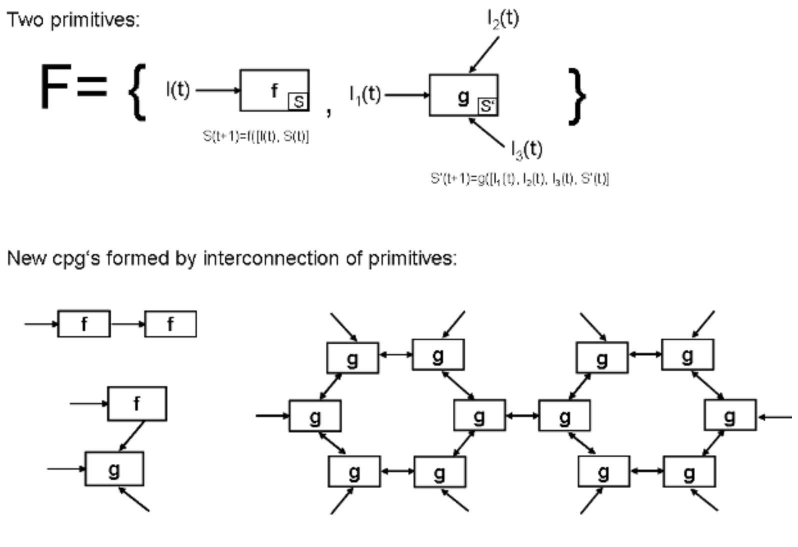

Two primitives:
F= {
I(t)
f
S
S(t+1)=f([I(t), S(t)]
I_1(t)
I_2(t)
I_3(t)
g
S'
S'(t+1)=g([I_1(t), I_2(t), I_3(t), S'(t)]
}
New cpg's formed by interconnection of primitives:
f
g

Figure 57. Primitive and complex constrained generating procedures (gpc).
(based on Holland, 2000)

Each of the nodes of P3 could be a gpc. The states of nodes are their positions. The inputs for each node are the states of other nodes (i.e., their position). Node x_3 could interact by having nodes x_1 and x_2 as input: the rule of gpc x_3 states that its distance to its predecessor (x_2) should be roughly the same as between x_1 and x_2. Hence, if x_2 is located 10 units on top of x_1, then x_3 has to be located roughly 10 units on top of x_3. A relocation of x_3 would have impact to all other elements in order to preserve the structural quality. Changes of parts feed back to the whole structure.

We can consider complexes of gpcs as a gpc of a higher level. This is similar to the decomposition of a complex system into nearly independent but still connected parts as suggested by Alexander (1964) and Simon (1996).

3.2.10.4 Pattern generators

A similar approach is developed in *Elements of Pattern Theory* (Grenander, 1996). Building blocks are called generators g_1, g_2, g_3 ,…g_i. To represent a specific pattern, a set of generators is needed – the generator space G. A generator can create one part of a specific structure, e.g. a point in a coordinate system, the position of a house in a neighbourhood, or a specific fragment of a class diagram. However, a generator does not only define what it creates but also the bonds to its environment b_1, b_2, b_3, … b_w.

Equipped with a specific set of generators from generator space G, we can "[..] glue generators together and the bonds will tell us what combinations will hold together. This is a bit like chemistry: atoms (generators) are connected together into molecules (configurations) and the nature of the chemical bonds, ionic, covalent, and so on, decides what combinations of atoms will be stable enough to form molecules" (Grenander, 1996, p. 83).

The bonds of each generator g describe the valid environments in which it can be used. We can define the bond value space B that contains all bonds of all generators. If we cross the bond value space B with itself, then we have the set of all possible connections between all bonds of all generators. Its elements, pairs of two bonds, can said to be regular or irregular by assigning a truth function:
$p: B \times B \rightarrow \{TRUE, FALSE\}$

We can say that a configuration of pattern generators is regular, or fits internally, if all its internal connections of bonds are regular. The external bonds of a configuration define the boundary between the form described by the generator configuration and the context of other generators. To achieve fitness between a local configuration and a surrounding configuration, each of the external bonds has to connect to a bond of the environment in a regular way. A complex configuration can be parted into a sub configuration, and it is possible to combine regular configurations to larger configurations.

A specific instance of a structure that is generated by the generators of G is called an image. A specific setup of generators can be considered as a template that generates the "same" image again and again. For example, each of the various instances of P3 is an image of a regular combination of generators. In our original world structure this configuration of generators has been a template because it produced exactly the same structure again and again. Such a template allows no variations within its own space dimension. This is like the shape of a stamp that creates the same picture again and again. However, there may be variations in other dimensions as one can use different colours for the stamp.

3.2.11 Implications for design patterns

We have seen that not all realizations of forms are meaningful and efficient. Design patterns are not just any arbitrary recurrent part of design. However, meaningful parts are found on different levels of scale and they often overlap. The closure of a pattern has impact on its frequency. Recurrent information allows the extraction of information as knowledge about a structure – information as knowledge. By means of abstraction two different forms can be considered as equivalent. Mapping slightly different structures to one generalized form reduces the complexity of the given structure. Knowing a small number of forms and patterns can help to understand the complete structure. However, the process of abstraction omits the actual details of the structure – that means a loss of information. Therefore, we discussed which information we can afford to omit in order to keep both the emergent meaning and the gestalt of a pattern as a whole. A pattern should capture what is universal to all its instances. This universal can be found in both concrete and abstract forms. It can shine through clearly in models (both abstract and concrete). Abstractions are at risk to loose the essential character and abundance of a universal whereas single exemplars are at risk to obscure the perception of a universal for they manifest often many features at the same time. The differentiation of a universal into a multiplicity of forms shows parallels to the unfolding of patterns when symmetry is broken. It is important to not only focus on the resulting pattern but also on their generation processes.

3.2.11.1 The meaning of recurrence

We have also shown reasons why and when the splitting of a pattern into smaller sub pattern makes sense: when a sub pattern frequently recurs in other configurations as well. However, such an extraction is not prescriptive. A solution can be described directly in all its details or it can be partly expressed in terms of other patterns. The latter reduces redundancies in pattern descriptions but requires that all sub-patterns are also described properly.

We have also reasoned that sometimes redundancies in pattern descriptions occur because parts of patterns can overlap. Such redundancies can be necessary because each pattern needs to stand on its own. For example, if one element plays a role in several patterns, then each of the pattern descriptions should integrate that element.

3.2.11.2 Patterns are relative to their world structure

Which parts of a structure recur as a whole and their frequencies depend on the considered world structure. The patterns that we have found for our world structure are only meaningful in that world. If we consider an extension of our world structure, we could find other recurrences, structural qualities and hierarchies that make more sense.

The larger the world is, the more confidence we can put into the identified patterns. If our world structure would indeed be the world we live in, we could only be confident in the patterns observed if we knew the whole world – which is impossible.

In two world structures (e.g. different domains, different faculties, different programming teams) different sub-structures can recur. Also, if we extend a world structure (e.g. we learn from other domains, other faculties, other teams or other cultures) the most efficient distribution of patterns and sub-patterns might be different. It is important to understand that the efficiency of a pattern is always relative to the scope in which it was mined.

3.2.11.3 Patterns, case studies, models and examples

In general, each pattern description is more efficient than describing single designs, because it covers multiple designs. Since multiple design instances always contain variations, we need to abstract from the single instances. Abstraction always means a loss of information and we have discussed which kind of information we can afford to loose. We have argued that we can abstract from details but that we need to preserve the structural qualities of a pattern. One way to achieve this is to construct or use an existing model. The class diagrams of software patterns are rather models than abstractions because the more universal structure of a pattern manifests in the concrete class diagram of a pattern. For example, the class diagram of the OBSERVER pattern has the same structural quality as a class diagram of an actual implementation; the details change – such as function names, parameter types, number of methods etc. – but the structure is preserved. The interactions between the pattern elements are also universal between all instances of a proper OBSERVER.

The diagrams found in *A Pattern Language* are representations of the patterns that preserve the implicate order of elements participating in their structure. That is why diagrams are so important: they carry the essence of the pattern implicitly and their roughness allows the adaption to specific situations.

In educational scenarios, a good model is often an exemplary implementation of a method. A model of a BRAINSTORMING is a simple performance of the method without mixing it up with other methods. Educational pattern descriptions very often use a photograph to represent a model situation.

We have also seen that patterns of different levels of abstractions are possible. More abstract patterns (or form categories) contain more members. However, we have less information about the actual form of a particular member. More specific patterns cover less parts of the world but they provide more information about their actual form. The reason is that more specific patterns have already unfolded more of the structural qualities.

3.2.11.4 Different realities

Our excursion about universals suggests that the recurrent structures and their gestalt qualities are real. However, we have also seen that there are many realities (e.g. different ways to organize a structure). Some organizations are plainly false, others are less efficient.

We can never be sure whether we have found all the recurrences in a closed world. If we discover new patterns, previously observed patterns may be less efficient. Moreover, the efficiency – or appropriateness – of patterns can only be judged objectively if we consider the whole world structure. Yet people have made different experiences and each person has only seen some parts of a world. An expert is better qualified to perceive stable patterns for s/he has seen a large part of the world in her or his domain. More experience is analogue to the extension of our example world structure. The more experienced an expert is in a domain, the more s/he has seen of the world of that domain. Hence, someone with a lot of experience is better equipped to identify stable patterns. If we enter a new domain we may soon identify patterns. But to be sure that these patterns are adequate we need to have a lot of expertise.

Of course, the world experiences of different persons overlap – we are all living in the same universe. But how we construct our own patterns is a psychological question that will be discussed in the next section.

3.3 Patterns in our heads[20]

"People have had patterns in their heads

for as long as there have been heads."
John Vlissides

3.3.1 Motivation

Because design patterns are derived from real world designs it is assumed that the patterns are true as well (DeLano, 1998). The trouble is to ensure that a pattern has been captured in the right way. There is no automatic or formal procedure to mine a pattern. Patterns are fuzzy, and people may have different patterns in their head. To illustrate this, consider the concept of a LECTURE as a pattern. Most people understand what is meant by "lecture" and have their individual ideas what is meant by the term. The exact and intrapersonal essence of context, forces, problem, and solution is hard to capture, however. Some individuals may focus on a lecture's organizational issues, others may be more concerned about didactics. The point is that people have different patterns in their heads. As a result, the documented patterns would not only vary in style but also in content depending on the authors who wrote the pattern. The fact that patterns are part of the human mind is outlined by Vlissides (1997): "[…] people have had patterns in their heads for as long as there have been heads. What's new is that we've started naming the patterns and writing them down."

Sometimes, pattern languages have been considered as a mental model that a designer has (van der Veer & Melguizo, 2002). However, we propose that schema theory offers an alternative explanation of how patterns are represented in the mind and how they are acquired. Like patterns, a problem schema "allows problem solver to group problems into categories in which the problems in each category require similar solutions" (Cooper & Sweller, 1987, p. 348). A problem schema, too, captures the invariant parts of a problem and has slots that can be filled with the specific properties of the problem (i.e. schema instantiation). As in patterns, there is a generic solution procedure attached to the problem representation, which is executed to produce a solution to the problem (VanLehn, 1989). That patterns can be used to support students in the process of schema induction is pointed out by Muller (2005): "Defining a pattern is a way to make explicit the common features of analogous examples. The act of defining a pattern may be viewed as introducing a 'hard copy' schema in order to enhance the formation of a cognitive schema". Mislevy et al. (2003) also consider design patterns as guiding structures or schemas.

3.3.2 Introduction to schema theory

"The captain asked the passengers to fasten the seat belts. They were ready to take off." Now, after reading the previous sentences did you have the image of an airplane at the airport in your mind? How is that as the text never mentioned an airplane or an airport? Obviously the propositions in the text allowed you to interfere a situation that is more complex and richer in its details. This comprehensive situation is an interrelation of events and entities, and is stored in an internal data structure that can be activated by recognizing its typical features. Such data structures, or schemas, are mental representations in an individual's mind. Very different types of information can be stored in schemas, including physical objects, plans and strategies, behaviour patterns, or design knowledge. A schema bundles all the experiences within one class, that is, distinct experiences with similar features, and offers a generalized or abstracted representation. This process has been termed schema abstraction (Gick & Holyoak, 1983) or schema induction (Cummins, 1992) in the respective psychological literature. It is characterized by extracting features that are shared by the experiences within a class and that are thus relevant to the schema as well as by abstracting away from irrelevant or superficial variations of these experiences. Thereby the resulting generalized representation allows to integrate even slightly different new experiences within the same data structure.

In the following we will summarize the various aspects and flavours of schema theory while being aware that different research groups have quite different assumptions how schemas are achieved and function. Thus, schema theory is rather a theoretical framework that is specified in several ways.

[20] An earlier version of this chapter has been published at PLoP 2008 as "The relations between design patterns and schema theory" (Kohls & Scheiter, 2008).

3.3.2.1 Scientific origins of schemas

The term "schema" goes back to Plato and Aristotle (Marshall, 1995). In Plato's dialog *The Meno*, for both concrete and abstract concepts, a schema refers to the essential commonalities for example in music or shapes, or even in what makes up a brave man. Similar, Aristotle speaks of form (or schema) when he means the essence or nature of the thing, that is, which basic properties and characteristics make an object distinct. In that sense the meaning of schema is already similar (if not equal) to our understanding of what a pattern is, since patterns always try to capture the core and invariants of concepts. For example, Buschmann, Henney & Schmidt (2007, p.3) refer to the patterns an experienced expert can draw on as 'solution schemes': "Expert architects and developers can draw on a large body of these 'solution schemes' to address design problems that arise when developing new systems and maintaining existing ones."

In this discussion we pay attention on how these schemas or patterns are represented in our mind. For Kant, a schema was the link from empirical information to the pure categories or concepts that he believed were given a priori in the mind. Every perceived stimuli had a relation to an ideal concept (see previous section on universals), for instance the perceived chair can only be interpreted as a chair if it is linked to the idea of a chair. This linkage is done through schemas, which are representations of the perceived phenomenon. While psychologists keep the idea that perceived information is linked to pre-existing knowledge in the mind, they no longer believe that this knowledge is given a priori. Rather their interest focuses on the processes of how schemas are acquired, applied, stored and manipulated in memory. The term was established in psychology by Bartlett (1932), much research has been done by Piaget (Piaget, 1952; Piaget, 1971; Piaget, & Inhelder, 1969) and Rumelhart and Norman (1978). Schema theory has been introduced to education by Gick and Holyoak (1983), VanLehn (1989), and Sweller, van Merriënboer & Paas (1998). In the educational context, problem schemas are seen as the pivotal mental structure that are constructed by learning from examples and that guide problem solving in scholastic domains like mathematics or physics.

3.3.2.2 Schemas as structural units

A schema is a structural unit that represents a concept, situation, event, plan, behaviour etc. in a generalized form, that is, it contains an abstract representation of multiple instances of the same kind. In schema theory this unit is an internal data structure in the memory that organizes the individual's similar experiences. It is used to recognize similar and discriminate dissimilar new experiences, access the essential elements of the commonality (both verbal and nonverbal components), draw inferences, create goals, develop plans and utilize skills procedures, or rules accordingly (Marshall, 1995). For example, the schema of a CAR is a generalization of all the cars a particular individual has seen or experienced before. Though it does not contain all the details of any car seen, it contains the essentials, the core features and properties shared by (almost) all cars. If an individual sees an object that shares the same elements and relations as stored in his or her CAR schema, s/he will recognize it as a car. As such, the car appears as a *gestalt*, an encapsulated unit that appears as a whole form, in which the configuration of components is in full harmony. Gestalt theory is ubiquitous in schema theory and pattern theory. Both Bartlett (see Hesse, 1991) and Piaget (see Scharlau, 2007) have been inspired by gestalt theory. Alexander explains our perception of whole forms with reference to gestalt psychologists (see section 2.6.2.1).

To get a better understanding of what holds a schema together as a unit, in the following three structural features of schemas will be outlined. First, schemas are constructed based on variables and their associated values, whereby second, choosing precise variable values will lead to a concrete instantiation of a schema. Third, schemas can be used to represent declarative as well as procedural knowledge.

3.3.2.3 Variables and variable values

In schema theory, the building blocks of schemas are thought of as variables (sometimes called slots or attributes; Anderson, 1983), whereby each variable can take a (constrained) range of variable values (or slot fillers or attribute values) as input. The possible space of variable values reflects the range of experiences a person has had with a specific concept and determines what new experiences will be right away accepted as belonging to this concept, that is, without further need of modifying the existing schema. In some cases, especially if a schema is highly specific, a constant may be used as a filler rather than a range of variable values. For instance, for many people a car always has four wheels, whereas for others, who have made a broader range of experiences, the attribute 'number of wheels' may take values from 3 to many. As will be outlined in section 3.3.5, learning in schema theory often refers to modifying the variable values by either extending or constraining their possible range.

The variables that make up a schema are often interrelated, which is also why the configuration of their variable values is not independent. Imagine a DRIVING schema, where the CAR is a variable value as one can drive other vehicles than cars. The DRIVING schema may have other variables, such as driving style, street type, or target. Each of the variables may be filled with different values, for example the variable slot street may be filled with highway, country road, or jungle road. If one drives on a jungle road the options for driving styles are reduced as the ground does not allow speed rallies. Hence, variables are constrained by each other, and a specific configuration of some variable values implies that certain values are more likely for other variables. These constraints and probabilities for the occurrence of specific variable values are an implicit part of the schema structure. They have two important functions (Rumelhart & Norman, 1978):

1. The range of valid objects for a variable slot is given.
2. If specific information about the variable is missing, it is possible to make guesses taking the probabilities into account.

For example, there are different types of vehicles that can be driven; that is, a car, a bus or a balloon are all valid values for the DRIVING schema. However, one cannot drive a traffic light – which would be an invalid value for the slot. Also, the valid values are of different likelihood. In the sentence "Bob was driving yesterday", the concrete vehicle is not mentioned. However, it is more likely to think that Bob was driving a car than to imagine Bob in a balloon, although this is a valid option. In the same way the introductory example activated the schema of an AIRPLANE AT AN AIRPORT because it is most likely that the information given refers to that situation. There are other situations that could be meant, for example a speedboat making a trip with tourists or a child in a role play. Which values are probable is hence determined by individual expectations based on one's own experiences, or heuristics.

Variables and constraints are fuzzy rather than exact. Not every constraint has to be satisfied in order to accept a concrete instance as being represented by a schema, but not too many constraints may be violated either. For example, a car usually has four tires. But if a tire is missing, it still remains a car. However, if the form is too different from former experiences of what a car is, we will no longer perceive a car.

To summarize, schemas are organizational structures of memory that interrelate variables that frequently reoccur. A design pattern, too, is a logical structure that consists of variables (Alexander, 1964). The variables are highly interrelated and changing one design variable to make it fit to the requirements of the design problem can cause other variables to misfit (e.g., using high quality material to make a machine robust may increase the production costs too much). While the interrelations of variables within a pattern are strong, the interrelation of variables between distinct patterns is loose (Alexander, 1964, p. 64). Hence, the configuration of variables within a pattern does not affect other patterns and thereby reduces the complexity of the design problem.

3.3.2.4 Schema instantiation

Filling the variable slots with exact values is called schema instantiation and its result is a specific mental representation of the abstract concept stored in the schema (Rumelhart & Ortony, 1977). For example, while the CAR schema consists of the abstract representation of all cars, its instantiation refers to a specific car. Therefore, schemas also have a generic component that allows the individual to derive specific structures from the abstract structures. The idea of a Table can be differentiated to various concrete tables. Not only can the individual retrieve the images of cars s/he has seen in the past by instantiating the slots with the right information, s/he can also imagine new cars. Or, to use a plan as an example, once the child has learned how to drink from a cup (it has acquired that schema), it can transfer that knowledge to any other cup or cup-like beverage containers. A schema understood in this way is not just a collection of experienced objects but an object space that allows to recognize similar objects and mentally (re-) generate objects. Likewise design patterns are not only generic but generative as well. In the way schemas form configuration spaces, patterns define design spaces that include all the designs that implement the pattern.

The generation space of schemas is defined by the variable slots and their associated value constraints, including variables that are mandatory (that is they nearly always occur) or optional. By a specific configuration of the variables, objects that are experienced (i.e. recognized) or have been experienced in the past (i.e. retrieved from memory) are represented in the schema instantiation. Since the variables slots have different probabilities for their values, the object space implicitly includes a prototypical representation of the object. This prototypical representation is the schema instantiation in which all the slots are filled with default values (this is similar to the model of pattern P3 in section 3.2.7.3).

In the way schemas constrain the valid values, patterns capture the invariants of good designs. Because for any form the list of characteristics is infinite, one has to extract the variables that matter (Alexander, 1964), i.e. the ones that are critical to distinct one form of another.

3.3.2.5 Declarative and procedural knowledge

The data structures can capture declarative knowledge as well as procedural knowledge. Some authors distinguish between scheme (procedural knowledge) and schema (declarative knowledge), and some translation errors have blurred this original differentiation by Piaget. However, the selection, instantiation, modification, creation and adoption of schemas apply equally for scheme and schema. Thus, in the ongoing text we will refer to both types of knowledge by the term schema.

Because schemas can represent both, objects or operations, a schema that consists of sub schemas can include both declarative and procedural knowledge. Both kinds of knowledge can be either directly included in a schema or referred to by variables. Because each schema has always optional external relations, it will point to various types of associated knowledge. Of special interest is the structure of problem and solution which will be discussed later.

In analogy, patterns capture declarative knowledge and try to externalize procedural knowledge in their solution sections since patterns are both a process and a thing. Furthermore, a pattern tells about a form not only what it is but also what it does (Alexander, 1964).

3.3.2.6 Interrelations among schemas

Every schema is embedded in a hierarchical network that reflects the interrelations among schemas. These interrelations can take two forms, namely, (1) part-of relations that reflect the fact that each schema always is an ensemble of interrelated parts that co-occur in a network, where each of the parts is a schema itself (Rumelhart & Norman, 1978), and (2) is-a-relations that introduce the idea of inheritance of structures into schema theory. Both types of interrelations are outlined in the following paragraphs.

3.3.2.7 Has-a/part-of relations

The variables are the means of interrelated entities in schemas. A set of networked variables builds up a schema, but each of the variables represents a schema itself. For example, the CAR schema has a set of strongly coupled sub schemas such as WHEEL, TIRES, MOTOR etc. A schema can be completely decomposed into its sub schemas, and each of the sub schemas has a part-of relation to the superior schema. Vice versa each schema is always a sub schema of a larger context. The TIRE schema for example can be part-of the CAR, but it can also be part-of an AIRPLANE. Similar, a CAR can be part-of a HOLIDAY TRIP or a DRIVE TO WORK. Thus, not only the internal elements of a schema are variable but also its external interrelations to superior schemas. In general, each schema A consists of multiple sub schemas $B_1 \ldots B_n$. The schema A, however, is more than the sum of its sub schemas because it itself defines additional structure concerning the way in which $B_1 \ldots B_n$ interrelate. Also, each schema is part of a larger whole C. As a general rule we can note, that schemas can be embedded into other schemas recursively similar to patterns.

Due to the aforementioned part-of relations activating a schema in memory will give access to two types of information by means of association. On the one hand, because the variable values of the current schema are schemas themselves, further information on these variable values can be retrieved from memory. On the other hand, because a schema may itself be part of other schemas, information on the latter can be accessed in memory.

Schemas can also be loosely coupled. For example, the schema MAINTENANCE is associated with a car, but if one sees a car one does not always think of maintenance. If, however, the car is rusty or makes suspicious noise, the association with maintenance is more likely; that is the variables *condition* and *sound* influence the activation of the schema MAINTAIN CARS which consists at least of the schemas CAR and MAINTENANCE. Hence, the association between MAINTENANCE and CAR is operationalized by the superior schema MAINTAIN CARS. Strong coupling, as found by the subparts of a car, is also given for CAR and DRIVING because both schemas are in most of the cases slot fillers for the DRIVE CAR schemas.

This hierarchical organisation of schemas is considered by some schema theorists as the complete cognitive structure of human acting (Eckblad, 1981). It is a means-end structure in which an initial state is transformed into an end state by splitting up each problem into sub problems. At each level of the hierarchy there are alternative schemas possible to solve the problem. “Another scheme might be a tentative plan for the solution of a problem, which is characterized by the start conditions, an outline of the goal to be reached, and some ideas of the route of subgoals by means of which one will try to reach the goal” (Eckblad, 1981, p. 19).

3.3.2.8 Is-a relations

This type of relations between schemas is given by different levels of abstraction. One can have a schema of a DOG while having a more specific schema of a POODLE, or a more general schema of an ANIMAL. If a person perceives a poodle, it can be represented by any of the three schemas appropriately. However, the schemas vary in detail and generality. While the POODLE schemas only applies to a small subset of animals, it provides more specific knowledge about poodles, for instance what colour the fur can have. The ANIMAL schema on the other hand applies to all animals but comprises only information that is valid for all animals. It is possible to switch the level of detail by which we consider an object. We can see the poodle as a poodle, a dog, or an animal. Schemas that are at the same level of abstraction, however, are competing: a dog is either a poodle or a dachshund, but never both. For any actual situation, schemas of different levels of abstraction are valid, but for any level of abstraction there will be one schema that is more appropriate than others. Similar to inheritance in object oriented programming, the more specific schemas inherit the properties and procedures of the more general schemas. Although not explicitly stated in the literature on schema theory, we can assume that multiple inheritance takes place since concepts can overlap. For example, a family dog may inherit the properties of dogs but also the properties of the schema family. Furthermore, we can say that the type is an important constraint for the variables of a schema. If a specific schema type is accepted in a variable slot, then the specialized schemas are valid as well. For example, in the DRIVE schema, there is the slot STREET. STREET itself is a schema, and we can fill the slot with any type of street, such as HIGHWAY, COUNTRY ROAD, or JUNGLE ROAD. These streets differ in their specific properties, for example surface materials, size or number of lanes, but on a more abstract level they are all streets and share the common emergent properties of a street.

Like schemas, patterns are always generalizations leaving out the details. They abstract from concrete forms. The level of abstraction is given by the number of features and relations that are taken into account by a schema or pattern. Schemas can also exist at all scales. The hierarchic organisation of patterns is reflected in pattern languages (Alexander et al., 1977) and logically explained by set theory (Alexander, 1964, p. 78). The relationships between patterns have been classified in various ways. Noble (1998) distinguishes between the *primary* relationships that "a pattern *uses* another pattern, a pattern *refines* another pattern, or a pattern *conflicts* with another pattern" and the secondary relationships (i.e. used by, refined by, variants, variant uses, similarity, combines, requires, tiling, sequence of elaboration) which can be expressed in terms of the primary relationships. It is notable that in Alexander's pattern language (Alexander, 1979) the only expressed relations are the uses/used by relations at the beginning and end of each pattern. This type of relation delegates problem-solution details to sub patterns and it shows that Alexander's emphasis is on the decomposition of complex problems. In fact, some of his patterns are related in different ways. HOUSE FOR SMALL FAMILY, HOUSE FOR COUPLE and HOUSE FOR ONE PERSON describe different house types. On the one hand, these patterns can be considered as conflicting patterns (Noble, 1998). On the other hand, one could argue that they are different refinements of an abstract pattern - because the houses are specialized solutions addressing particular contexts.

Comparing the schema and pattern concept, the Has-a/Part-of relations of schemas are *uses* relations in patterns, and the Is-A relations in schemas are *refines* relations in patterns. So, what about the *conflicts* relation? In a way, one could argue that patterns never are in real conflict because different contexts readjust forces for the same problem type and therefore require specific solutions, e.g. the different house patterns all apply to different contexts (families, couples or singles). Likewise the problem of transportation calls for different solutions (e.g. car, bus, train, plane) because of varying contexts and hence different weights on the forces (Corfman, 1998). If the problem statement is to create a website, the solution may look quite different depending on whether one creates a news site, a university portal, or an online community (van Duyne, Landay, & Hong, 2004). In schema theory the *conflicts* relation is not an explicit relation type between schemas. Nevertheless, it is assumed that schemas compete for activation depending on their appropriateness as the next section shows.

3.3.3 Activation of Schemas

To understand a given situation and to plan appropriate activity means to select an appropriate configuration of schemas taken from the currently available repertoire of an individual. The selection of appropriate configuration of schemas to account for the situation is called comprehension by Rumelhart and Norman (1978). This configuration consists of schemas that can be instantiated in a way that the current empirical data fits into slots of the activated schemas. The process of activating the right schemas is a complex pattern matching task in which memorized schemas cooperate and compete (Schnotz, 1994). Schema activation is triggered by the existence of relevant data. For example, the data "eyes" and "nose" can activate the schema HUMAN FACE, because it has the appropriate slots in which the given data can be properly assimilated. However, on a more abstract level there are competing schemas, such as ANIMAL FACE. Also, to activate a more specific schema MALE or FEMALE FACE, further information is needed. A name, "Sarah", could clarify that the schema FEMALE FACE should be activated. The three activated schemas EYE, NOSE and FEMALE NAME hence cooperate to activate the higher level schema

FEMALE FACE. Schemas are considered to be self evaluating. That is, for any given input stimuli a schema can evaluate the likeliness that it can represent the empirical situation. The process of calculating such a fitness value is executed by the schema itself – the algorithmic knowledge to generate output values according to input values is implicit given in the structure of the schema. To recognize the situation, the schema that best matches the features of that situation is selected. This is a fuzzy logic operation because the variable slots of the schemas are assigned with probabilities. So, if in the current stimuli some features are missing, unusual or additional, a schema can still be appropriate to represent the situation if in total the features match is better than in any other schema. Some feature settings, however, are not tolerated. For example, if the information "diameter of 5 cm" comes in, the schema HUMAN FACE has to be discarded.

3.3.3.1 Bottom-up and top-down

The process of schema activation works both bottom-up and top-down (Schnotz, 1994). Perceived information activates low level schema candidates at the bottom of the hierarchy, which produce output data on a higher level. For example, visual patterns activate specific shape schemas in this bottom-up process. This compound information is the input for higher level schema activation. If a schema is found in which the configuration of basic spatial objects is similar to those of an eye, this schema is activated and the output data is "EYE". This output data, again, is available in the recognition process. In combination with further data, such as "NOSE" and the word "Sarah", the FEMALE FACE schema is activated. From this schema, additional knowledge is inferred in a top-down process. For example, the schema knows that lips, ears, eyebrows etc. are most likely to co-occur. This knowledge can then be used to further interpret the available data on lower levels. In a comic drawing we can interpret a single line as a mouth if we have the context information that we see a face. This inference also helps to substitute missing information. If a text does not mention a mouth explicitly, we derive a default assignment from the MOUTH schema and fill the slot of the FACE schema. Hence, we are likely to think of red lips, although green or blue lips are possible (if a lipstick is used).

The context of concurrently activated schemas and the current empirical perception are both input data. The later is encoded into schemas on the lowest level. A specific combination of lower level schemas makes it more likely for a specific schema on a higher level to be activated. This is due to the interrelations of variables. If the current lower level schemas are "better" slot fillers for schema A than schema B, then schema A will be stronger activated than B. As a result, the activated schema A raises expectations about further variables. That is, on a lower level additional schemas become more likely to be found because they are expected from A. For any schema X, the activation depends on the available data. Positive stimulation can come from lower levels (X has parts that are activated) and higher levels of the hierarchy (X is part-of an activated schema). The strength of activation influences other schemas in their own evaluation. If, for some reason, the schema HUMAN FACE is denied (too many mandatory parts may be missing), this has consequences for other schemas that are already activated. The denial of the FACE schema is negative input for the EYE and NOSE schemas, hence, they have to be re-evaluated. Maybe the shapes that first had been perceived as eye and nose have a different meaning.

3.3.3.2 Activation triggers output

The evaluation of a schema yields a certain level of activation, which in turn will affect the recognition of an observed object or scene. Strong activation means the individual has recognized a familiar situation. With its activation, all the interrelated variables get weight. Some of these variables can be operations, actions, and plans. Hence, the recognition of a certain situation also activates elaboration, planning, and execution knowledge. Again, this knowledge can be split up into sub schemas. To achieve a required state (as defined by planning and execution knowledge), the schema refers to lower-level schemas that have stored the knowledge how to achieve that state. On the lowest level, the schemas cause actions of the individual. Throughout this process, the schemas are constantly re-evaluated and if at one point there is a misfit, strategies have to be changed, that is errors have to be corrected. The schemas itself and the cooperation of schemas (which are defined by the interrelation) can be improved by continuous learning (tuning).

3.3.4 Problem schemas

In problem solving, two situations can be distinguished, namely, solving problems for the first time and handling recurrent problems (VanLehn, 1989). If a specific type of problem is encountered for the first time, the problem solver does not have any knowledge (no problem schemas) available to solve this problem. Thus, s/he can implement only problem solving strategies that do not require any prior knowledge, that is, so called knowledge-lean problem solving strategies. While these strategies may fail if dealing with rather complex tasks, they are the only ones available to the problem solver. On the other hand, if the person has already acquired knowledge on solving problems of a particular type, because s/he has made recurrent experiences with this problem type – either by learning by doing or studying illustrating examples – then knowledge-rich strategies become available. These strategies rely on applying existing problem schemas to the task at hand. Both problem-solving strategies will be outlined in the following paragraphs.

3.3.4.1 Solving design problems for the first time

Problem solving based on applying knowledge-lean strategies can be analyzed by considering two cooperating sub processes, understanding and search (Newell & Simon, 1972). The understanding process produces mental information structures that represent the problem according to the understanding of an individual. Its major output are two states, the current situation in which the problem arises as the initial state, and the desired situation in which the problem is solved as the final state. Solving the problem is a search process in which the solution is calculated or found by taking moves in a problem space. The problem space consists of (1) the initial problem state, (2) operators that can change a problem state into a new state, and hence allow moving in the problem space, and (3) the goal state. Each state in the problem space can be reached by a sequence of operator applications (VanLehn, 1989). The difficulty of the problem is correlated with the topology of the problem space (Newell & Simon, 1972). States in the problem space could be formally described by a state-representation language. For design problems, this language could describe the modelled form of the design and how it satisfies the given forces. If a state is found in which the form balances all forces, this state represents one solution. During the search for the solution, the problem solver may not only find that s/he gets closer to the solution if s/he is on the right path. Some of the insights gained are often changes of the problem space itself (Duncker, 1945; Ohlsson, 1984). For design problems this could mean that during the process of finding the right form, the designer finds additional forces that have to be taken into account. A state in the problem solving space corresponds to a set of assertions (VanLehn, 1989). In the case of the designer an assertion means that a particular configuration of a variable has influence on the forces. Moving in the problem space is to vary settings of design variables to see whether the whole design better fits to the problem in the context. The operations of the problem-solving process then are the incremental changes to the assertions, that is the piecemeal variations of form. The challenge is that each variation influences more than one force and that some changes may lead to unwished results. Hence, the operations can lead to dead ends. In that case, a backup-strategy can lead the designer back to a state which is closer to the solution. S/he can then proceed with another set of operations (that is change design variables). Heuristics, or rule of thumbs, are often used by problem solvers to decide which operations to apply in a state that satisfies certain conditions related to the heuristic. Finding a good solution is not random search. Rather, it is achieved by small transformation of states in a problem-space. The transformations are chosen based on experience of the individual. If the steps taken mean progress towards the solution, then the path is continued and the applied heuristic is strengthened. Finally, a solution state is reached.

To handle complex (design) problems, these can be decomposed into sub problems of smaller granularity. The hierarchical decomposition of artificial structures and problems is described by Simon (1996). It is notable that Alexander has referred to some of the earlier works of Simon (Alexander, 1964, p. 74) who has done much research on problem solving and is often referred to in literature on problem schemas. That the patterns in software design are likewise an approach to decompose complex systems into a small number of recurrent subsystems as described in *The Science of the Artificial* (Simon, 1996) is highlighted by Grady Booch (1998).

3.3.4.2 Recurrent problems and solutions

If subjects recognize the stimulus of a familiar problem, however, they do not seem to start the search process through the problem space. Rather, they retrieve a ready-to-use solution procedure and follow it. The knowledge unit that is available to solve problems of similar classes is called a problem schema (VanLehn, 1989). Problem schemas are more likely to be applied by experts of a domain and allow for the implementation of knowledge-rich problem solving strategies. They consist of both, knowledge about the problem class (i.e. declarative knowledge) and the skills to solve problems belonging to that class (i.e. procedural knowledge). They are considered a specific form of schemas, which have the same characteristics as have been discussed in the previous sections. Thus, both parts of a problem schema are composed of variables that can take different values, which are used to represent a problem class' relevant features as well as operations to solve respective problems. Moreover, as suggested by schema theory problem schemas are interconnected in a hierarchical network by part-of and is-a relations.

The first part of a problem schema comprises knowledge of the problem class and its constituent features, which is important to recognize and activate the appropriate problem schema. In that way, a problem schema serves to better understand the problem. Experts usually have a better understanding of a given problem in their domain in that they can represent the problem based on its second order features rather than its first order features (Chi, Feltovich, & Glaser, 1981). Second order features seem to be coherent with the structure of the problem, in opposition to first order features, which are coherent with the surface structure. The second order features are the ones that allow deeper elaboration and understanding (Marshall, 1995). Once a problem schema is recognized and triggered, its solution procedure can be applied to the representation of the problem. The solution itself is a structure that has slots, which can be filled by arguments that are taken from the problem representation. As discussed in the previous sections, the slots allow for variations, for ranges of acceptable values. This is important, as it allows to adapt the generic parts of the problem schema to the specific operands given in a problem.

3.3.4.3 Subdivisions of problem schemas

Marshall (1995) who has researched problem schemas for mathematical problem solving, suggests that a problem schema consists of identification knowledge, elaboration knowledge, planning knowledge and execution knowledge. Thus, she splits the problem and the solution structures into sub structures. Though this split up is plausible, it is rather arbitrary. Does every problem schema require execution knowledge? Think again of the car, which can be the solution to travel from A to B. Once we realize that a car is a proper solution, what is the proper execution knowledge? Do we have to know how to build a car, or only how to drive a car? Is having a driving license a precondition to let the car be a proper solution? What about asking a friend to drive? There are other ways to subdivide problem and solution. In the context of design patterns the problem is usually split up into the core problem, the context, and forces. Each is an interrelated sub structure of the problem structure. Indeed, one can further differentiate the perspective on the context as there are several types of contexts (available resources, required skills, cultural habits etc.). The solution, too, has different sub structures, including the form of the solution (a car), the process of creating the solution (design and fabricate the car) and how to use the solution (driving the car). This nature of problem schemas can be summarized as follows:

1. A problem schema is a single schema that relates a problem to a solution.
2. Both, problem and solution are sub schemas that are integrated into the problem schema.
3. Both, problem and solution can be further sub divided into sub schemas.
4. If the problem schema is activated, then it automatically triggers the solution schema as a whole.
5. The solution schema activates its related substructures such as planning, execution, elaboration, use forms etc.

The problem schema is the body which integrates all the sub structures into one schema. Each of the structures is an abstraction of concrete instances in which the structures have been found – a pattern. It is remarkable that in the schema literature, the expression "pattern" is often used only for the identification task, i.e. the pattern recognition. Of course, the assigned procedures and plans are patterns as well as they capture the invariants of the solution. The combination of the problem pattern and the solution pattern is captured in a problem schema. Thus, the design patterns of an individual are equivalent to problem schemas since design is considered to be a problem solving task in the pattern community. To be more precise, design patterns are a special case of problem schemas as there are other types of problems.

Both, design patterns and schemas can have relations to other concepts (patterns or schemas). By the combination of several schemas/patterns, composed schemas/patterns result. Therefore, the patterns of problems can be linked to the patterns of solutions. The patterns of contexts, values, consequences, execution knowledge, elaboration knowledge, forces etc. can also be linked. Since each schema defines a configuration space, problem schemas provide a linkage from a problem space to a solution space in the very same way as design patterns do.

Problem schemas apply to knowledge-rich rather than knowledge-lean problem solving (VanLehn, 1989). Similar, design patterns apply to design problems that require a large body of information and specialist experience that should be available to the designer (Alexander, 1964). As with design patterns, problem schemas can be utilized in combination to solve larger and complex problems. If a certain combination of problem schemas reoccurs, it is likely that this combination establishes a new schema with the participating schemas as variable slots. Which leads us to the question that has been elegantly excluded so far: how are schemas acquired?

3.3.5 Schema acquisition and development

According to Piaget (1952), schemas are created and adopted by two interplaying processes: assimilation and accommodation. Assimilation is the integration of external elements (perceived stimuli and the data available from the output of other schemas) into developing structures or into already existing structures. In the assimilation process, properties of the external element that are incoherent with the activated schema will be ignored. However, not every feature can be skipped. Thus, to integrate the current stimulus, the existing structure has to change according to the special properties of external elements. This process is called accommodation. If a schema has assimilated many experienced elements and the accommodation process has established a logical structure that is capable of representing all of them, then the schema becomes stable. It is in a state of equilibration in which it is resistant against outliers and interferences.

3.3.5.1 Assimilation

Assimilation is the application of subjective schemas to represent a perceived situation, or perceived objects. Which schemas are appropriate to represent the situation has been discussed in the section "Activation of Schemas" (3.3.3). Since schemas are abstractions based on previous experience, the empirical information of the currently perceived elements is unlikely to fit completely to the activated schemas. In the assimilation process the information is adopted so that it can be integrated to the activated schemas, that is irrelevant features or superficial variations are ignored. The structure of the perceived object is altered so that it can be represented by schema instantiations. This implies that our perception of things depends on our previous experience and what we expect about situations. For this reason, optical illusions let us perceive things that are not really there. This fitting of perception to expectation does not only apply in everyday situations but also in scientific scholarship, for example new findings are tried to be harmonized with existing mathematical concepts or specific models and theories.

The assimilation process can be expressed by this formula (Piaget, Fatke, & Kober, 2003):

$$(T + I) \rightarrow AT + E$$

T is the existing structure, I is the integrated element (the perceived stimuli), E are the eliminated components (the structural parts of the stimuli that are ignored in favour of the schema) and A is a coefficient > 1 that expresses the strengthening and addition to the existing structure.

As an example consider the simple schema structures A and B (spatially represented) and the example stimulus in figure 58.

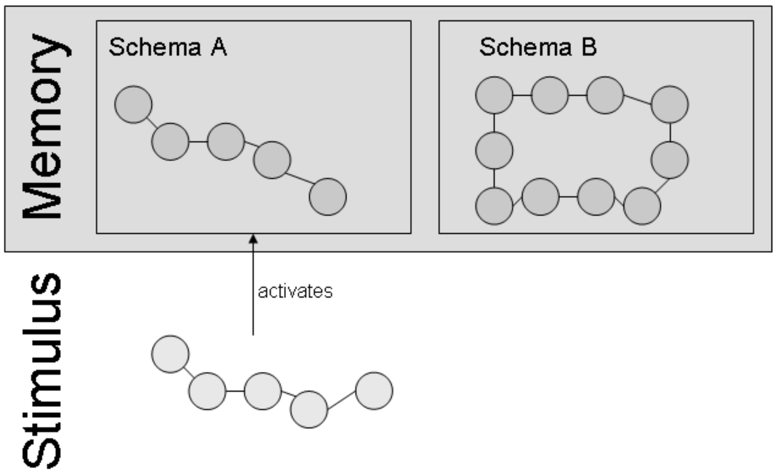


Figure 58. A stimulus is assimilated by the best fitting schema and activates it.

Though the stimuli is not exactly represented by schema A (the most right component, or variable, has a different position, or different value) it matches better with A than with B. Therefore, schema A is activated and assimilates the stimulus. If the stimulus is fully assimilated by A, then its minor difference is ignored. That is it will not be stored permanently in memory; rather, the given stimuli is represented by A. Assimilation is conservative and tries to subordinate the environment into the organism (Piaget, 1975). How exactly the specific experienced stimuli can be retrieved (reproducing assimilation) depends on the specialization of schema A. If A is a very specific schema then it prescribes a detailed structure; hence, it has to be very similar to the perceived stimuli in order to assimilate it (recognizing assimilation). However, if A is on a more abstract level, then it will accept a wider range of stimuli. If the stimulus is assimilated by such an abstract schema, less exact details are available for retrieval since A generalizes over some features (generalizing assimilation). This explains why story fragments that represent typical situations are retrieved by subjects in terms of general information rather than the specific information given in the text. For extraordinary events, however, subjects often remember the specific details (Bartlett, 1932).

Besides the differentiation between reproducing, recognizing and generalizing assimilation, Piaget distinguishes between simple and reciprocal assimilation. Simple assimilation means the integration of external elements into an existing schema (activation of a single schema). Reciprocal assimilation means that a schema integrates sub schemas (has-a relations), or multiple schemas (is-a relation).

The integration of new data structures without changing the existing schemas is referred to as accretion of knowledge by Rumelhart and Norman (1978). Though they do not speak of assimilation, the proposed process of accretion is very much alike. It is also based on schema activation, or as the "natural side effect of the comprehension process" (Rumelhart & Norman, 1978, p. 45). In their model, the specific situation or experience is stored in data structures, which are instantiation of appropriate schemas. To retrieve information of a particular experience, the instantiated schemas are used to reconstruct the original experience. This instantiation allows the storage of episodic knowledge while the schema contains the generic structural knowledge. The model differs from Piaget's assimilation process in that specific data is stored separately in instantiated data structures. Thereby it is better suited than the Piaget model to explain how one can remember specific details that are distinct from the generic structure. However, it does not explain why specific information is lost (forgotten) or replaced by generalized knowledge. The parallel to assimilation is that existing schemas are required to interpret and memorize new input, the new experience is associated with a configuration of schemas, and that the structure of the schemas does not change. The creation and evolution of schemas is credited to a different process: accommodation (Piaget, 1971), or tuning and restructuring (Rumelhart & Norman 1978).

3.3.5.2 Accommodation

If there was only assimilation, no learning would occur because there would be no process of changing or restructuring in the knowledge structures**.** Accommodation is the counterpart of assimilation as it adopts the schemas to be coherent with the new experience. If an external element is assimilated not all of its special features can be discarded or ignored. So, the assimilation schema has to be changed in a way that it can represent the specific new experience as well as the older ones. The requirement that older experiences of the same class must remain instantiatable limits the process of accommodation, however. This limit is expressed by the term A in formula $(T + I) \rightarrow AT + E$. The new schema structures are a cognitive adoption of former schemas.

Analogous to assimilation, Piaget distinguishes between simple and reciprocal accommodation. Simple accommodation happens whenever a schema is activated. Reciprocal accommodation happens if multiple schemas are assimilated by a superior schema, and the coordination of the sub schemas is adopted.

Rumelhart and Norman (1978), too, distinguish two types of schema development: tuning is the evolution of existing memory structures (that is of a single schema); restructuring is the creation of new ones (e.g. create a new ensemble of co-operating schemas, or copy an existing schema for modification).

Tuning only affects an adjustment of variables, their values ranges, and probabilities for both values and co-occurrence of values.

The constant and variable terms can be changed in four ways (Rumelhart & Norman, 1978):

1. The accuracy is improved by having more differentiated constraints.
2. The applicability is generalized by extending the range of acceptable variable values.
3. The applicability is specialized by limiting the range of acceptable variable values, in the extreme by replacing it by a constant.
4. Default values can be established if instances of the schema happen to be frequently assigned with typical values.

As an example, consider how a schema accommodates as a number of similar stimuli are assimilated into the same schema:

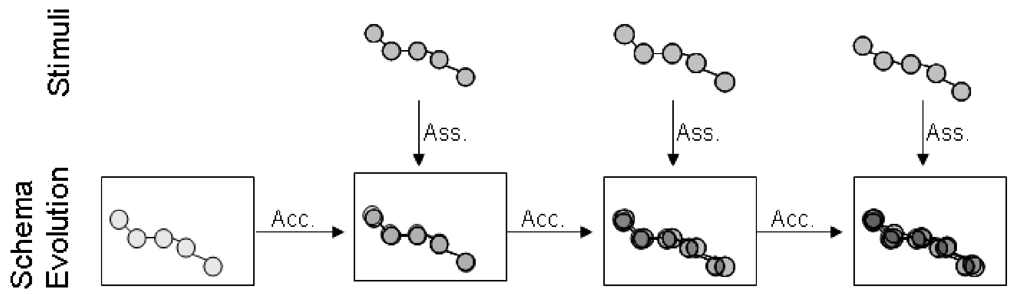


Figure 59. Evolution of a schema by assimilation and accommodation

The number of components (or variables) does not change in the evolution process. However, the ensemble changes in a way that the configuration space extends. The positions of the components (i.e. the variable values) get a wider range. Some configuration values are more probable because they occurred more frequently (in the figure, the strongest memory trail is where the stimuli overlap, thus marking a default but not a mandatory configuration). Also, the relational structure does not change. The schema remains one schema, and its interrelations do not change.

Another type of learning happens if the existing schemas are restructured. In this process new schemas are created either because the new experience does not fit to the currently available schemas, or its organization is not satisfactory. Rumelhart and Norman (1978) name two types of schema creation as a result of restructuring: patterned generation and induction.

A patterned generation is a creation of a new schema by copying and modifying an existing one. As in the use of analogies, some variables are left out, others are added and some are changed in their values. For example, one can develop the schema of a rhombus, by getting the information that it has the same relationship to a square that a parallelogram has to a rectangle. Since the old schema is first generalized and then specialized, the logical structure of inheritance is implicitly given. One can say that this type of restructuring concerns is-a relations.

The other form of restructuring is schema induction. As it concerns the contiguity of schemas it will restructure has-a relations. The co-occurrence of certain configurations of schemas will generate a new schema that stores this formation. Though not mentioned explicitly by Rumelhart and Norman (1978), one can assume that schemas can not only be created by composition of sub schemas but that a schema can be decomposed into sub schemas as well.

To illustrate the two types of restructuring, consider figure 60 in which the schema evolution progresses by perceiving more stimuli.

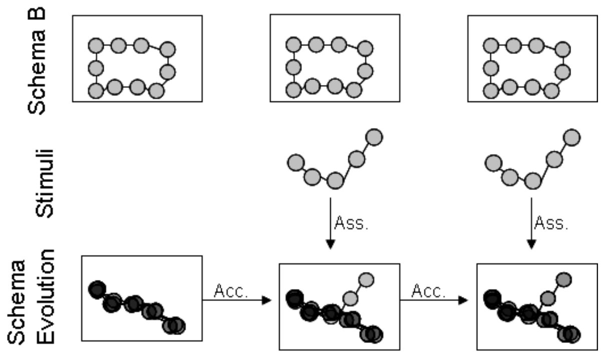


Figure 60. Continued schema evolution

Although the structure of the new stimulus is not so similar to the evolving schema, it is assimilated by it because it better matches this schema than the competing schema B. However, the evident differences in the last two stimuli require accommodations. The configuration space of the evolved schema is not satisfactory for the experiences it is meant to represent. Therefore, restructuring (or reciprocal accommodation in Piaget's terms) is required. The original schema can be copied and modified according to the specializations:

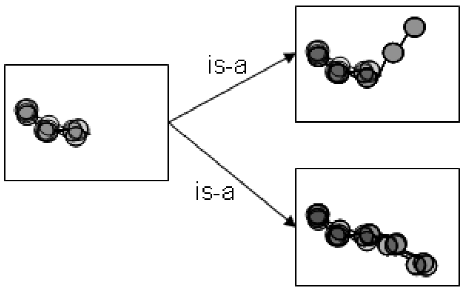


Figure 61. Restructured as patterned generation.

Schema induction, on the other hand, would create new schemas by composition or decomposition. In the given example, this would mean, that the schema is decomposed into three sub schemas (figure 62) that can be aggregated into two macro schemas (figure 63).

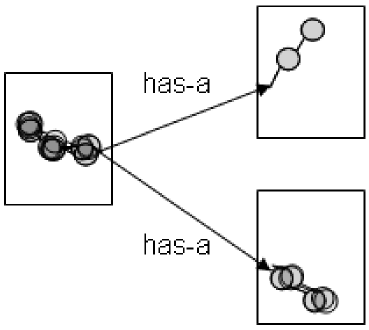


Figure 62. Three sub schemas.

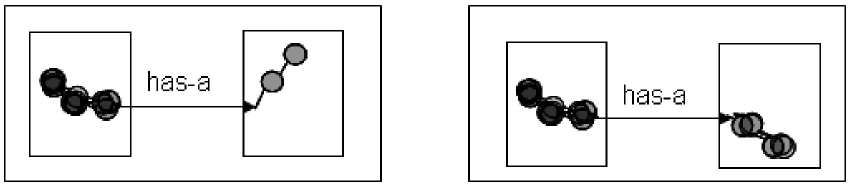


Figure 63. Aggregation into two macro schemas.

In the model of Rumelhart and Norman (1978), learning can occur without accommodation. Accretion of knowledge stores the special instances of experienced situation separate from the schemas and therefore does not tune or restructure the schema. In opposition, in the schema model of Piaget assimilation cannot occur without accommodation and vice versa because all knowledge is stored as schemas in the memory. Assimilation always comes first and the assimilation structures compete for activation. Accommodation is subordinated and forced by the fact that the assimilated structures must fit into the configuration space of the schema structures.

3.3.5.3 Equilibration

Not every new experience will completely reorganize the existing schemas. For example, if one sees a car with three wheels, the car can be assimilated by the CAR schema. However, the CAR schema will not be accommodated in a way that cars with three wheels are a common pattern. The reason is that with every integrated experience the probabilities for certain configurations are modified. If one has seen thousands of cars with four wheels, the one car with three wheels will not change the schema significantly, if at all. Such a stable schema is in an equilibrated state. The schema structure, that is the variables, their constraints, interrelations and occurrence probabilities, describes a configuration space that is capable to represent all members of a class even the extraordinary ones. Although schemas become stable and equilibrated, this does not mean that there is no change any more. One can assume that a schema is never fully equilibrated because after thousands of hours of practice, an individual can still improve his or her performance (Crossman, 1959; Fitts, 1964).

Because the schemas are built upon the experiences of an individual, no two persons will have exactly the same schema. Though each person may have a schema of a CAR, these schemas are not identical. They may be very similar but there are also differences. A person that lives in North America may have developed a different prototype of a car in mind than a European. The reason is that people in North America have developed different default values for the variables. Also, the first car one has seen may play and important role since it was the initial ground for the schema. Some people may have knowledge about how to check oil and test the air pressures while others lack this information. Some people may be concerned about environmental issues when thinking of cars while others see cars as status symbols rather than transport vehicles. So, even for cars there are many different schemas that may have had developed among individual peoples.

3.3.6 Related theories to schema theory

It is important to keep in mind that schemas are only a conceptual model of human memory; there are no physiological entities in the human brain that directly correspond to a schema. Rather, schemas are the structural units we can become aware of. Other theoretical approaches such as categories (Kelly, 1982), frames (Minsky, 1977), scripts (Schank, 1975; Schank & Abelson, 1977) or mental models (Gentner & Stevens, 1983) refer to the same memory structures but offer different explanations how the structural units are stored, related, created and instantiated. Categories are basic concepts (such as ANIMAL, DOG or CAT) whereas schemas can also represent more complex mental structures (Zimbardo, Gerrig, & Graf, 1999) and situations such as DOGS HUNTING CATS. Scripts are schemas that contain specific sequences of events in well-understood contexts, the classic example is the visit to a RESTAURANT. Scripts have been deeply discussed for the use in education and computer-supported collaborative learning (Dillenbourg, 2002; Dillenbourg & Jermann, 2007; Weinberger, Ertl, Fischer & Mandl, 2005; Kollar, Fischer & Hesse, 2006). The close relation to design patterns has been noted by Villasclaras-Fernández, Hernándes-Leo, Asensio-Pérez, Dimitriadis & Martínez-Monés (2011). They point out that design patterns can describe the context, problem and forces that justify a script. Mental models, too, can be generated by the interplay of appropriate schemas. What is common in all these approaches is that they provide conceptual models of classification. One can call the class of CARS or the BRAINSTORMING method either a schema, a category, a script, a mental model, or a pattern – depending on the theoretical framework you are in.

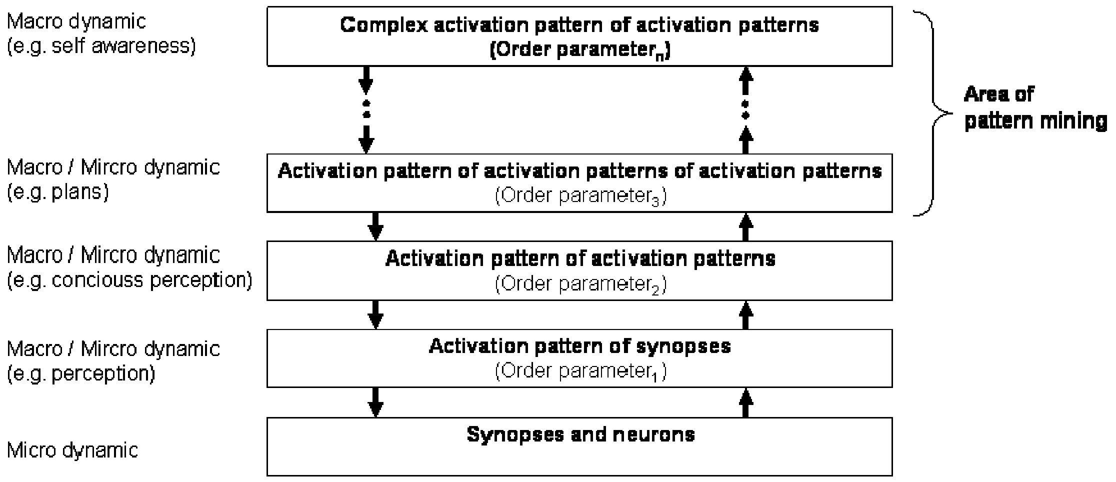

Macro dynamic (e.g. self awareness)
Complex activation pattern of activation patterns (Order parameter$_n$)
Area of pattern mining
Macro / Mircro dynamic (e.g. plans)
Activation pattern of activation patterns of activation patterns (Order parameter$_3$)
Macro / Mircro dynamic (e.g. conciouss perception)
Activation pattern of activation patterns (Order parameter$_2$)
Macro / Mircro dynamic (e.g. perception)
Activation pattern of synopses (Order parameter$_1$)
Micro dynamic
Synopses and neurons

Figure 64. Hierarchy of complex activation patterns.
Based on Mainzer (2008, p. 69), extended with a note at which levels pattern mining takes place.

Neuroscience goes beyond conceptual models since it tries to map conscious states to activity patterns of firing neurons. The fact that neuroscientists, too, speak of patterns highlights the importance of structural relation and co-occurrence of features in stimulus. "…, human brains operate fundamentally in terms of pattern recognition rather than of logic. They are highly constructive in settling on given patterns and at the same time are constantly open to error" (Edelman, 2006, p.82). Distant neurons will make synaptic connections if their firing patterns are temporally correlated. The synchronized firing creates neural clusters and patterns that correlate with mental states of an individual – the mind emerges (Metzinger, 2000). The existing synaptic connections influence the activation patterns that respond to a given stimulus while at the same time the synaptic connections are modified themselves (Lloyd, 2004). The conscious mental state is represented by the "integrated pattern of neural activity" (Edelman, 2006, p.92) and this activity depends on both the current stimuli and the individually developed brain structure formed by former activation patterns. This mixing of the past with the present stimuli to a phenomenal experience is conceptually described as assimilation in schema theory. The altering of the brain structure by strengthening synaptic connections between temporally correlated firing neurons can be considered as accommodation. Then, are the neural activity patterns instantiated schemas and contain mental representations of the design patterns as we know them? Not exactly because schemas and design patterns are located on a macro level whereas neurons and synaptic connections are on a micro level in a hierarchical network of activation patterns (Mainzer, 2008), see figure 64. In the same way the physical object of a car can be decomposed into its components and further into its molecular structure, the schema of a CAR is decomposable into sub schemas and finally into neural activity. In our conscious thinking we have only access to the upper levels. The lower levels can be captured by brain scans. However, today's research is far from linking the recorded brain activity to the actual conscious state. Due to the complexity of dynamic systems this may never be the case. For our enterprise, the mining of design patterns, we are interested in the macro patterns, anyway.

3.3.7 Implications for design patterns

At the end of the equilibration process, an individual has developed an ideal of the object in question. And in that, s/he has developed an attitude towards the ideal object, giving her or him the capability to judge whether the things or events in questions are the way they ought to be – whether they just feel right. This is true for non-design objects (such as trees), for emerging situations and structures (such as sunsets or the behaviour of animals in the woods), and for deliberately designed objects (such as cars, or a software systems) certainly as well. Having an ideal concept of a thing lets one judge the beauty of it. From a philosophical perspective, Kant (Kant & Erdmann, 1880) argues that we perceive objects as beautiful when they are good projections of the ideal form. Sometimes things intuitively feel beautiful. In this case the judgement depends on our implicit design knowledge, stored in equilibrated problem schemas that evolved from the experience of seeing things that worked or not. In other cases, the elegance of design is less obvious and we are required to evaluate the values and benefits consciously. We have to reflect all the known facts relevant to a design rather than depending on intuition alone (Alexander, 1964). At this point the explication and externalization of problem schemas in the form of patterns becomes a helpful tool to reduce the complexity (allowing to reason about more forces), to share information and to ensure that the patterns of an individual are also perceived by other people.

3.3.7.1 Patterns as subjective structures

The crux with patterns is that they depend on schemas constructed by the individual. The more empirical experience is given for an individual, the more stable his/her schema gets. If one has experienced 100 exemplars, s/he is more likely to judge by experience which parts of the configuration are invariant and which change from exemplar to exemplar. Likewise, an expert is aware of commonalities and differences, s/he has all the patterns, abstract and precise, in her or his mind. “A man who knows how to build has observed hundreds of rooms, and has finally understood the ‘secret’ of making a room with beautiful proportions” (Alexander, 1979, p. 222). This expertise allows to create new things by copying good designs. For the domain of architecture, Alexander states (1979, p. 207): “A pattern language is really nothing more than a precise way of describing someone's experience of building.”

The recognition of patterns depends on the cognitive skills of an individual. Even if we systematically analyze design artefacts and define criteria to classify similarities of the objects, these are individual judgements. Formalizing does not work because we are not only interested in a pattern of form but in the pattern of *form as a solution*; only as such it becomes design and has a meaning. It resolves the forces of a problem in a specific context. There is a difference between the form of a hammer, and the semantic of a hammer as a tool that can be used in certain situations to solve a physical problem. The semantic is something that goes further than the form itself and it requires a human being to make a meaning of the hammer. Therefore, an algorithmic approach must fail. Alexander appreciates that fact when he dispenses from his proposed *Program* as a method to find pattern diagrams and realizes that the human capabilities allow to find the patterns in a more natural way: “If you understand the need to create independent diagrams, which resolve, or solve, systems of interacting human forces, you will find that you can create, and develop, these diagrams piecemeal, one at a time, in the most natural way, out of your experience of buildings and design, simply by thinking about the forces which occur there and the conflicts between the force” (preface of 1971 edition of Alexander, 1964).

This view is very coherent with schema theory. In the same way that the interaction of assimilation and accommodation leads to an equilibrated schema, the attempts of explication and externalization are evolutionary processes. An implicit schema emerges by experiences and to be stable it has to properly represent the object space it refers to. Based on the implicit schema, an individual can create an explicit mental representation of it in the pattern format. For example, one can implicitly know that a car is a good solution for a problem in various situations. To consider a car in the dimensions of context, problem and solution creates an explicit view on the car as a design pattern. This view again will be adopted several times to ensure that it is coherent with the implicit schema it is based on. If a first stable version is available in the mind, one can start the externalisation process by writing the pattern. The written pattern description again will be adapted piecemeal to be coherent with the explicit mental view on the design pattern. Indeed, this process is not a one way path. Writing the pattern will take influence on the mental representation of the design pattern. Reasoning about one's own problem-solving strategies creates new experiences. Thus, the implicit problem schemas may change as well in this process.

3.3.7.2 Loss of information (and reality)

The trouble is that a lot of things can go wrong, because our minds may be the best pattern recognition apparatus but they are not perfect. First of all, in any transformation there is loss of information. The richest in-form-ation is available in the formation of a single design artefact itself. Even an objective pattern (if we could grasp it) has to skip a lot of details because it is an abstraction of all the artefacts that manifest the pattern. While the (hypothetical) objective pattern forms a specific design space of all designs of its class, the subjective schema of an individual is only based on a finite subset of this class. This leads to the slightly different ideas of cars and their uses depending on where we live. Furthermore, the mental image of a single object (e.g. a car) is always only a projection of the object itself. Likewise, the explicit mental pattern is a projection of the implicit internal schema and the written description is yet another projection. Each projection means loss and potential sources for failures. While the loss of information can simplify things and make them easier to handle, the danger is to loose critical information, that is to ignore design elements that matter. Failures are even worse because it means that an individual has constructed a mental representation of a real world pattern (or some parts of it) in a wrong way.

One of the pitfalls in objectively evaluating the correctness of a pattern is the individual judgement whether a design is good or bad. The human mind builds up schemas of all kinds of recurrent structures it perceives; including horribly bad designs. The pattern community calls those recurrent designs anti-patterns or dysfunctional patterns. "Some patterns have recurrence but not quality. These are bad patterns" (Buschmann, Henney & Schmidt, 2007, p31). But dysfunctionality is not always a matter of objective judgement. Everybody agrees that software that can be easily adapted to new requirements is better than software which lacks this capability. But not everyone agrees that PowerPoint adds value to the way people present information – it is a matter of personal preferences and the quality of presentations one has experienced in the past. In fact, many of the patterns in "A Pattern Language" are related to values that depend on individual attitudes. For instance, not anybody may agree that a world government, as proposed in the first pattern INDEPENDENT REGIONS (Alexander, 1979), is part of a good design. It may be, but people have different judgements on that issue. Such different world views are the reason for heatful debates in democratic systems.

In the pattern community, people often trust in the validity of a pattern because it is assumed that it has been written by an expert. The designer, however, is not satisfied by validity alone. Besides validity the designer expects usefulness and generative qualities from a pattern. A pattern can be scientifically valid by describing the structure correctly and analyzing the consequences – benefits and liabilities – adequately. However, the choice of the level of abstraction, granularity, and details influences the usability for designers. Indeed, some schemas only represent recurrent structure in design but not design patterns since they do not capture adequate solutions to problems. Some people see patterns where there are no such; this mild form of platonic schizophrenia has been anticipated by the pattern community (Marquardt, 2005).

As an example, let us reconsider the CAR pattern and see what could go wrong with the pattern mining, both informal and systematically. First, one could miss some important components by saying that in essence a car is four tires, a steering wheel, passenger seats, and a body. And indeed this configuration is a recurring structure in proven design solutions (the cars) and therefore formally a pattern candidate. The trouble is that this four-components pattern is incomplete. It misses important parts such as brakes, engine, fuel tank and many things more. Without brakes, a car is hardly good design. One may implicitly know that a car requires these components but fail to make this explicit. Second, one could claim components as part of the pattern that are actually not. If someone only observes cars with gear sticks, s/he may find that a stick is part of the car pattern. But today, cars with sticks are only a variation, there are also automatic cars. The same can be said for trunks. A car without a trunk is still a car, hence trunks are only an option. If the individual observer only saw cars on streets, s/he could argue that streets are part of the CAR pattern, too. But they are not: cars can exist without streets and streets do not only serve cars. STREETS are an independent pattern that happens to occur often in conjunction with cars using it. Because it is so comfortable to drive on streets, CARS ON STREETS is certainly another pattern. However, in the mind of an individual, cars and streets may be an inseparable unit. For this person, streets are always part of his CAR schema, the pattern in her/his mind. Third, finding an appropriate abstraction level is a challenge. One has to question whether there is a general class of cars, or, each specific car class should be treated as another pattern. For example, although a race car and a family van share many properties, there are significant differences. And what about busses, trucks, or ambulances? These are all different patterns, but they are all specialisations of the CAR pattern, too. Which abstraction level works best depends on the granularity suitable for the target group. If a pattern goes deep into the details, documenting each class as a single pattern is worthwhile. Otherwise one would have to enclose all the different variations into one long description. On the other hand, if the variation is only small, treating each variation as a new pattern leads to a pattern explosion that fails to serve the original idea of design patterns to communicate expert knowledge in an efficient way.

Finding a pattern on the right level of abstraction, granularity and details seems to be very hard, in particular because each individual constructs his or her own patterns in the mind. The good news, however, is that these individually constructed patterns do not differ that much. Otherwise an individual would not know what is meant by the word "tree". Though individuals have different schemas of trees, they are not all that different. In an experiment (Kohls & Uttecht, 2009) we tried to find out whether people actually find the same patterns in design. The outcomes are reported in section 4.3.

3.3.7.3 Knowledge transfer

The goal of pattern writing is to exchange expert's knowledge. In schema theory it is assumed that knowledge is stored in structural units, which are basically patterns in one's mind. The pattern format therefore seems to be a very adequate vehicle to capture these nuggets of wisdom. It is the form of representation closest to the way knowledge is organized in memory. They can, therefore, most efficiently guide the reader to construct her/his own similar schema. It is important to note that, of course, a schema itself cannot be transferred. A pattern description only sketches the schema, but the understanding and capability of applying the pattern must be constructed by each individual him/herself. To support this process, illustrative examples are needed. Constructing a stable pattern requires experience. The more exemplars an individual has perceived, the better his or her schema will be. For this reason, examples play an important role in documenting a pattern since they can clarify the invariants and variations of design. Although a pattern explicitly captures the core of design, multiple examples are needed to induce a schema by a learner (Quilici & Mayer, 1996), thus, letting him/her explore the core by him/herself. Iba (2012) has developed a concept to learn a pattern language by personal communication: A large group of students gets sheets with brief pattern descriptions. The students highlight which patterns they have experienced and which they want to learn about. Studends who know an example for a pattern share their story with students who do not know the pattern yet. This way the concrete cases of a pattern are discussed in personal communication rather than in written form. Alexander points out that each person must have his or her own version of a language in order to make it a living language: "A living language must constantly be re-created in each person's mind" (Alexander, 1979, p. 338). Pattern descriptions may speedup the creation of the schemas but do not replace the process.

3.3.7.4 Mental construction of patterns

Because schemas are actively constructed and depend on the exemplars one has experienced, the schemas that represent the patterns of the real world may be different structures for each individual. The same is true for patterns. That there are different views on the same patterns can be illustrated by pattern descriptions of several authors that refer to the same pattern. The Gang-of-Four patterns (Gamma, Helm, Johnson & Vlissides, 1995) have been re-written many times in other books. There are programming language specific descriptions (e.g., Cooper, 2000), heavily illustrated versions (Freeman, Freeman, Bates & Sierra, 2004), and even patterns for dummies (Holzner, 2006).

Besides different views on the same patterns, there are many variations of a pattern, and a pattern can always be split up into sub-patterns or be combined with other patterns to macro patterns. None of these operations does harm to the validity of the resulting patterns, however they may affect the usability. In understanding how patterns are induced by individuals, one can find better strategies of finding appropriate patterns to document. In the competition of schemas, an appropriate level of abstraction and granularity is usually activated: If we see a car we usually think of a car and not of a vehicle or a specific car class. So, if we want to describe a car it is probably better to describe it as a CAR and not as a variation of a VEHICLE.

Maybe the most important issue of this section is to realize that the patterns in real design and the patterns in our heads are not equal. In spite of the practical nature of patterns, a pattern is only a theory as we will see in the next section. The next section considers how the pattern community has established methods and processes to evolve subjective pattern impressions into more objective pattern descriptions.

3.4 Explicit pattern descriptions as theories about patterns[21]

There is nothing more practical than a good theory.

Kurt Lewin

In theory, theory is the same as practice, but not in practice.

Fnord Bjørnberger

3.4.1 Motivation

Whether or not patterns can be called scientific methods has filled the beer cellar of Kloster Irsee with heated discussions among "practitioners" and "researchers" during many EuroPloP conferences. The practitioners usually reject a scientific approach to patterns, arguing that good patterns contain "nothing new", but capture existing knowledge. From their point of view, the nature of patterns is a specific and very useful genre for technical documentation. Unsurprisingly, pattern researchers beg to differ. They consider pattern mining (i.e. the discovery or explanation of a pattern) as a scientific endeavor. Patterns reveal previously unreported regularities. In the following sections, we try to reconcile both views, by distinguishing patterns that represent scientific progress from patterns that are just another – albeit effective – genre for documentation.

If patterns truly contain "nothing new", the question is: What is the point of stating the obvious? In fact, a pattern should not report on surface properties but rather "capture hidden structure" at a "suitably general level" (Coplien, 1996). To reveal such previously unreported structure certainly is a creative act. The argument that there is "nothing new" in a pattern must be rejected; otherwise there would be nothing new to physics either, since physical objects and the laws of physics have been around before, just as design objects or programming styles have been around before somebody sets up a pattern language or collection. The discovery and description of a new species is without question considered scientific progress. Of course, the animals are not new – they lived there before – but they are newly discovered. In the same way, "the most important patterns capture important structures, practices, and techniques that are key competencies in a given field, but which are not yet widely known" (Coplien, 1996).

However, not all pattern descriptions contain new findings. If somebody describes the OBSERVER pattern once again, this can hardly be called scientific progress – unless it contains new properties of the OBSERVER pattern which have not been covered before. Research in other fields can also contribute to the content of patterns. As Buschmann, Henney & Schmidt (2007) point out, patterns and pattern descriptions evolve over time. Including new findings, e.g. new relevant forces, new consequences, new contexts or limitations, means to get a better understanding of the nature of a particular pattern.

Also, each new implementation of a pattern in an appropriate context is a test whether the forces are actually resolved by the pattern or not. Such a test can succeed and provide evidence for the "truth" of the pattern. A test can also fail and potentially falsify the claims of the pattern. If a pattern constantly fails, it must be rejected. If a pattern continues to help those designers who apply it in the right situations and in the right manner, the pattern gets more reliable.

However, such corroboration must not be confused with ultimate truth. It is just a level of confidence we can put in those patterns that are helpful to the practitioner. And while patterns are all about practice we should not forget that they are of a theoretical nature nevertheless. In as much as physical theories are about the general physical laws of the universe, theories of practice are about the general rules of creating specific forms. While we can directly observe the single instances of design, e.g. a specific software design, we cannot directly perceive the generalized forms and the causes for a form as they are captured in a pattern. We might have seen "a force at work" but we have never seen a force directly. Hence, patterns are purely rational assumption, and they need, as any theory does, empirical evidence.

[21] An earlier version of this chapter has been published at PLoP 2009 as "Is that true? Thoughts on the epistemology of patterns" (Kohls & Panke, 2009).

In section 3.4.2 we will provide arguments why patterns should be treated as realized theories (abstract objects) about real things (concrete objects) rather than universal truths. We will approach the nature of discovering theories and patterns in section 3.4.3. The specific methods of pattern mining will be discussed in section 3.4.4. Section 3.4.5 discusses the level of confidence we can put into the methods and their results, the mined patterns, suggesting that written patterns often need further corroboration. The final section discusses the difference between general (theoretical) designs and specific (implemented) designs and the required skills in creating a design.

3.4.2 Patterns are theories

What we have learned from constructivism is that the apparent "objective reality" and the subjective meaning, value and volition are not a completely different kettle of fish. The observer is always part of the story and patterns are not neutral descriptions. Similar to the communication researcher Paul Watzlawick (1976), who posed the question: "How real is real?" we can ask: "How real are patterns?" Patterns have to be open to methodological falsification, since humans tend to find patterns everywhere – even in randomized circumstances. Consider the following example taken from Watzlawick (1976): People are presented with a multi-armed bandit to participate in a problem solving experiment. Their instruction states that they have to press a certain numeric pattern to successfully activate a buzzer. What the subjects do not know is that the reward buzzer works totally randomly; nothing they do influences the buzzer. During the first part of the experiments, subjects receive a certain percentage of random rewards. During the second part, they receive no rewards whatsoever. In the third and last stage, they are rewarded every single time. At this point, all subjects are convinced they finally found the "pattern" of successfully operating the multi-armed bandit. Even after the experimenter tells the truth about the setup, most participants find it hard to believe, claiming they found a regularity the experimenter was unaware of.

Likewise Feynman (1974) warns us to not only consider positive examples of a hypothetical assumption because this could lead to what he calls "cargo science" (an allusion to the anthropological concept of "cargo cult"): "In the South Seas there is a cargo cult of people. During the war they saw airplanes land with lots of good materials, and they want the same thing to happen now. So they've arranged to imitate things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head like headphones and bars of bamboo sticking out like antennas - he's the controller--and they wait for the airplanes to land. They're doing everything right. The form is perfect. It looks exactly the way it looked before. But it doesn't work. No airplanes land. So I call these things cargo cult science, because they follow all the apparent precepts and forms of scientific investigation, but they're missing something essential, because the planes don't land" (Feynman, 1974).

As these examples show, our perception of patterns is open to delusion and what we find depends on what we are looking for. It is therefore necessary to be able to distinguish between "correct" and "incorrect" patterns. This means that we have to investigate their content empirically and we have to make sure that the structure of a pattern is logically sound.

Design patterns are prone to the same misinterpretations of reality as the next examples show. In our collection of patterns for interactive graphics we mined several button forms to dynamically show and hide graphical elements (Kohls & Windbrake, 2008b). Technically there is no big difference between showing/changing a graphic on mouse click or mouse over events. We initially mined one pattern out of it. However, in an experiment we asked teacher students which interactive graphics they classified as similar (Kohls & Uttecht, 2009). Not one of them considered the two trigger events as similar in spite of the same reactions and graphics. Even more alarming, when we wrote the two distinct patterns it turned out that they were appropriate to different contexts and resolved different sets of forces.

Schmolitzky & Schümmer (2008) have written an excellent language for supervising thesis projects. However, some of the patterns' instant solutions show a preference of the authors to use wikis whenever possible. For example, their DIARY pattern suggests "The easiest way to implement a diary is to write it as a shared wiki page." We have discussed this over with several educators and most said that an e-portfolio or blog system would be the most appropriate thing for this particular task. Hence, it is more of a personal view than a definite truth which solution works best.

3.4.2.1 A pattern is not the same as its manifesting artefacts

Discussing patterns as theories is not common in the community of practice that mines and documents patterns. On the contrary, there is a widespread argument on being suspicious of theories and traditional science. Coplien (1996) stressed that patterns are made of "Stoff – real stuff, not platitudes and theories" and Rising (1998) highlights that “patterns are not theoretical constructs created in an ivory tower, they are artifacts that have been discovered in more than one existing system”. Another common view in the community is the notion of patterns as not being artificial, having emerged over time. Patterns are not “created artificially just to be patterns” (Buschmann, Henney & Schmidt, 2007, p.8). Similarly, DeLano (1998) observed: “Patterns are not grown or created. They are present in the artefacts that already exist.” Besides being made of recurrent real stuff, patterns are considered to be true or proven: “good patterns are those solutions that are both recurrent and proven” (Buschmann, Henney & Schmidt., 2007, p.14). Patterns capture “well proven design experience” (Harrison, 1998), and they are “tried-and-true” (DeLano, 1998). Authors like Schümmer and Lukosch (2007) consider existing examples as proof of the patterns. Finally, there is nothing new or inventive about patterns, as they “are an aggressive disregard of originality” (Foote, 1997). Gabriel (2002) stated “software patterns are about describing what works and has worked well rather than finding new ideas”.

Yet, we claim that patterns should be regarded as theories, implying that their content is of hypothetical - not true, inherent or given - nature.

We are not opposing the idea that the substance or base of patterns is (or should be) “real stuff” that has been observed or experienced. Our point is that there is an important difference between a pattern and the objects that manifest the pattern. In the same way there is a difference between a theory and the objective facts that are explained or predicted by the theory. For example, Newton’s laws of motion are a theory and the actual motion behaviour is the “real stuff”. Of course, the laws of motion have always been at work and there is good reason to believe that people have been aware of them long before Newton was – otherwise the pyramids would not have been built (Berkun, 2007). However, Newton was the first who has systematically explicated the general law. In the same way, patterns do not invent or create the design solution they describe. Alexander (1979, p. 261) stated: “The task of finding, or discovering, such an invariant field is immensely hard. It is at least as hard as anything in theoretical physics”. What is important for our pursuit is what we can learn for pattern mining from the methods of discovery, advancement, failure, falsification or proof applied in theoretical physics and other sciences.

Patterns are abstract entities that can only manifest in real instances. We can draw a pattern or represent it in other ways but what we are actually doing is drawing a diagram, sketch or model. It is an interesting question whether or not there is a “true” pattern behind the descriptions we give when we are writing pattern documentation. In that sense, the pattern is just an idea, in the same way that specific triangles are not the same as the concept of a triangle. We cannot imagine the concepts themselves but only imagine them by exemplification – we cannot see “the triangle” (the universal) but only “a triangle” (an instance), we cannot see “the beauty” but only “a beauty”. In the Platonic world view, all truths exists a-priori and sensual data offers only fuzzy representations of the original ideas. Hence, there is a truly perfect form of a table and pattern mining might be a way to come close to this form.

Asking about the “truth” in patterns only makes sense if we assume that patterns are not “real a-priori” knowledge, but theories, that can be tested, supported or falsified. Throughout this section, we will argue that patterns cannot be taken from a Platonic sky; they do not represent an absolute “platonic idea” of an object or its “true nature” but offer a constructivist theoretical explanation that captures one reality of objects. As such, we use patterns to make sense of the day-to-day practice of design in a pragmatic manner.

Buschmann, Henney & Schmidt (2007, p.114) state that the process of writing a pattern and the outcome, that is the written document itself, are both part of the pattern: “In most cases moving from one form to another is largely a matter of rewriting and remining the pattern, which is a profoundly creative activity”. So, writing a pattern is a creative and original act in spite of the “disregard of originality”. This is not a contradiction since we are considering two levels here: a pattern is original and creative, but the phenomena we describe with the pattern are not invented. They do “really” exist, whatever this “really” means (see section 3.2.9 on realism).

3.4.2.2 The structure of patterns and theories

In his seminal work on patterns, Christopher Alexander repeatedly considers patterns as laws or hypotheses, i.e. as morphological laws:
“Each one of these patterns is a morphological law, which establishes a set of relationships in space. This morphological law can always be expressed in the same general form:

$X \rightarrow r(A, B, \ldots)$, which means: Within a context of type X, the parts A, B, … are related by the relationship r” (Alexander, 1979, p. 90).

More compact: “IF: X THEN: Z / PROBLEM Y” (Alexander, Ishikawa & Silverstein, 1968).

Because we do not know laws a-priori, we have to formulate hypotheses about the laws. And a network of well corroborated hypotheses or accepted empirical laws is exactly what the kernel of theories is (Bortz & Döring, 2002). Alexander first speaks of patterns as hypotheses that can be tested in *Notes on the synthesis of form* (Alexander, 1964). In *The Timeless Way of Building* (1979) he points out that the distilled invariants of a pattern become empirically vulnerable:

"We can ask ourselves, is it true that this system of forces actually does occur, with the stated context? Is it true that the actual solution, as formulated, really does resolve this field of forces in all cases? Is it true that the precise formulation of the solution is actually necessary: that any entrance which lacks this feature must inevitably have irresolvable conflicts in it, which will communicate themselves to people who pass through it?" (Alexander, 1979, p. 269).

Because each pattern contains a network of hypotheses (the forces that link the context form to the solution form), we think it is adequate to consider a pattern as a theory and not only as an isolated hypothesis. Theories describe, explain and predict facts and phenomena. They can inspire research in new areas, predict events and provide promising instructions for practical action (Westermann, 2000). These functions of theories are remarkably similar to the general traits of the pattern format:

- a pattern describes the form of **recurrent** solutions and their contexts of applications,
- a pattern **explains the reason** for this fitness in terms of the forces,
- a pattern **predicts** that in another context of a similar kind, the pattern will help to solve the problem,
- a pattern is **instructive** (not necessarily prescriptive) for practical design action,
- a pattern is a **three part rule** consisting of
  IF X (context) THEN Z (solution) BECAUSE of Y (problem).

3.4.2.3 Arguments for considering patterns as theories

In the way Alexander describes patterns they very much look like theories. Why does it still seem odd to consider patterns as theories? The concept of a theory and a pattern are not identical because patterns are specific types of theories.

1. Patterns are written and documented in a specific way. The pattern format is a text genre different from usual scientific documentation. Pattern documentations are written appropriately for a particular target audience. However, that is only a rhetorical difference.

2. As theories about practice, patterns have a very powerful implicit test mechanism – a pattern needs to be useful to the practitioner. They are not about all kinds of practices and social behaviours. Only those phenomena that are judged to be relevant and have a direct practical use are picked out. This selection, however, does not make them any less theoretical. It is just a normative value which patterns are worth to be captured.

3. Patterns seek to abstract from concrete design solutions on a medium level of abstraction: "A pattern is abstract because it approaches the problem at a suitably general level, although the solution may entail details. A good solution has enough detail that the designer knows what to do, but it is general enough to address a broad context" (Coplien, 1996). Every abstraction is a loss of information and therefore makes a thing theoretical. We can afford to loose information about surface structure as long as the character of a form category - its essential structure - is preserved (see also section 3.2.7.4 on implicate order). Structure preservation requires that the gestalt (whole form) of patterns must remain perceivable. Wholeness sets the limit for abstraction. Once we reach that limit (without going further) we have a reasonable theory. Many sciences suffer from going beyond that limit and it is therefore that they seem to be too theoretical. However, if a theory looses its connection to empirical data it is actually not too theoretical but simply wrong or meaningless.

4. Patterns seem to be very different compared to e.g. the law of conservation of energy. Theories, we think, are usually very general and abstract. For example, the laws of motion apply for all physical objects. A pattern, such as an OBSERVER, only applies in a specific context. However, the OBSERVER pattern should always work if applied in the stated context. The context limits the scope of a pattern and therefore reduces the amount of contained cases– it does not apply for *ALL situations in the world* but for *ALL situations covered by the context*. Such restrictions regard the empirical content of a theory.

5. The leading paradigm in the pattern community is that patterns are based on experience. That is, the hypotheses of a pattern are derived from real instances and not "out of the blue". It is a misconception to think of

scientific theories as something invented "out of the blue". A theory should always explain and predict a phenomenon of reality. There are indeed different ways how theories are developed. While Alexander's work does foresee rational inventions of patterns, the current paradigm in the pattern community is to build a pattern upon past experiences, best or good practices. "The empirical nature of patterns suggests that they should be grounded in real examples" (Buschmann, Henney & Schmidt, 2007, p.100). Usually patterns are inductively inferred from real examples. As we will see, the concept of induction has been critically discussed in the philosophy of science for at least two centuries.

3.4.3 Induction, Deduction, Abduction

Looking at patterns as specific theories means that we do not have to invent new standards for testing and justifying patterns. Instead, we can rely on tried and tested methods – the patterns of scientific inquiry and empirical methodology. The objective of scientific inquiry is to create knowledge. Knowledge, to-know-that and to-know-how, can be defined as justified beliefs about facts, models and theories about the world (Schnädelbach, 2002). Likewise, a pattern expresses generative rules (laws or regularities) for the design of artefacts. We believe that these rules are tried-and-true and therefore justified.

From an empirical perspective, all knowledge must be based on experience. The phenomena of interest have to be observed and measured precisely and then we can try to postulate theories that summarize and explain the facts. Hypotheses are the result of inductive inference, coming from the specific to the general, from the concrete to the abstract. Inductive inferences are potentially truth extending and are the only strategy to find new insights.

On the opposite side, rationalists claim that there are significant ways in which our concepts and knowledge are gained independently of sense experience. Hypotheses are not the result, but the origin for empirical inquiries which test them. In a deductive argument, the conclusion cannot extend the information given in the premises.

For this reason, inductive inference seems to be more attractive and inductive empiricism is indeed the approach we apply when we mine for patterns. Digging for "nuggets of wisdom" seemingly means to discover what is already out there. However, induction comes with some problems.

3.4.3.1 Induction

In science, enumerative induction is the process of inferring from a number of observed positive cases the properties of new cases. This can be done for a specific case (extrapolation: this design has worked several times, so it will work in this case as well) or for all possible cases (generalisation: this design has worked several times, so it will always work in similar contexts) (Westermann, 2000).

Strong Induction (Generalisation)	Weak Induction (Extrapolition)
A_1 is a B_1. A_2 is a B_2. A_n is a B_n. Therefore, all As are Bs.	A_1 is a B_1. A_2 is a B_2. A_n is a B_n. Therefore, the next A will be a B.

Table 1. Induction

Induction is also eliminative (Westermann, 2000): in the process of extrapolation or generalizing it is also assumed that there are no other forces which (generally) influence the observed relations. Hence, we can say that mining design patterns is enumerative and eliminative induction. Although a new specific design task will indeed introduce new aspects to the context, new forces and a modification of the specific configuration, we are assuming that the essentials captured in the pattern do not miss critical forces and do not contain forces that were only incidentally at work in previous design cases.

The benefit of extending the knowledge about specific cases to general cases is at the same time the biggest problem of induction. Because inference by induction extends the information content (e.g. the conclusion contains more than the premise) there is uncertainty about the truth of the conclusion because it is not logically necessary that the cases of the past imply cases for the future, i.e. what has worked in the past does not necessarily work in the future. A recent popular science publication exploring the problem of induction and its consequences can be found in the book *The Black Swan* (Taleb, 2007). Induction builds on the assumption that the observed phenomena are uniform and will behave similar in the future. Besides the fact that this uniformity might be of different degrees (e.g. physical objects might behave more uniform than design problems or human behaviour), the assumption that the universe behaves according to the principle of uniformity is a theory that unfortunately builds on induction itself. This problem of lack of rational justification for this principle cannot be resolved (Hume & Buckle, 2007). We can handle this problem pragmatically and accept that we can never be completely sure.

From a constructivist point of view, there is yet another problem with induction. Since meaning is not inherent in the objects and artefacts surrounding us, but rather actively constructed, we can turn any given set of data into a variety of different theories (or patterns). Consider the following example: "A husband believes that his wife dislikes to be seen with him in public. As 'proof' he describes an occasion when they were late for an engagement, and as they were walking briskly from their car, she kept staying behind him. 'Not matter how much I slowed down,' he explains, 'she always stayed several steps behind me.' 'That is not true,' she retorts indignantly. 'No matter how fast I walked, he always kept several steps ahead of me'" (Watzlawick, 1976, pp.62-63).

Whenever we capture regularities in the data we observe, we use personal "punctuations" to determine cause and effect, beginning and ending of a situation. Just like we cannot be sure who causes the maritial problems in the example given above, we cannot be sure if the pattern we observe is actually truly a whole part of good design – or if there even is a pattern outside our perception. Let us illustrate this for software patterns. Buschmann, Henney & Schmidt (2007, p. 149) use a different punctuation for the ADAPTER pattern than the Gang of Four does: "Another example of two patterns hiding within a single pattern description is the case of OBJECT ADAPTER and CLASS ADAPTER, which are both contained within the description of Adapter [GoF95] […] Although they share a common intent, their complementary nature is reinforced by a dual narrative that runs through the pattern description". If only one of the two views is correct we have to either reject ADAPTER or reject the complementary pair OBJECT ADAPTER and CLASS ADAPTER. However, if we understand that the punctuations are not real a-priori but rather made up by the human mind to find meaning, then we can indeed accept both views. However, the two views may differ in efficiency and accessibility.

Furthermore, patterns not only build on the observable structure, but on further assumptions about the functional or even causal relations between objects. Functional arguments like "X works because of Z", "Z is there because it causes X" always contain an explanatory level that is not inherent in the data. Software design patterns are not just the recurrent class diagrams (the data) but also the forces (the reasoning). This indicates that patterns are theory-laden because forces are not directly observable.

3.4.3.2 Deduction / Falsification

Since we cannot prove whether a theory (or a pattern) will work in the future or in each and every case, verification remains impossible. Theories cannot be empirically verified (as positivists had thought) but only be tested. If they pass the test we can put more confidence into a theory. If theories consequently fail a test, we should reject a theory. This is the core idea of Popper's critical rationalism (Popper, 1972). To resolve the problem of induction Popper argued that science does not rely on induction, but exclusively on deduction, by making modus tollens the centerpiece of his theory. It has the following argument form:

If P, then Q.
$\neg Q$
Therefore, $\neg P$.

Science is gradually advanced as tests are made and failures are accounted for. To be tested, a theory (respectively the contained hypotheses) must be falsifiable. That is, it must be possible that if a theory is tested, it could fail.

The same should be true for patterns. We should not assume that patterns are true or that we could provide ultimate proofs in a mathematical sense for patterns. A pattern based on proven design does not imply that the pattern itself is proven. Rather the proven designs provide evidence (not proof) for a pattern. What qualifies a specific design to be proven is another question. However, even the strongest tests for existing designs do not imply that we have generalized the pattern appropriately. And even if so, we cannot be sure that it will work in the future.

Corroboration of theories should not rely on past data because there is the danger of making up a theory that just fits the previous cases. Patterns that are mined based on "real stuff" must necessarily be formulated in a way that they account for previous cases. But do they hold for future cases? This we do not know. The degree of corroboration is higher if we test a pattern for new cases and we must allow the option that patterns can fail. That makes a pattern empirically vulnerable: if we apply a pattern in the right context but it does not solve the present problems it is indeed falsified. That patterns can fail in principle is a good thing because it is the only way to test them. To which extent a theory can be tested depends on its empirical content (see section 3.4.5). To which extent a theory is actually tested and evaluated is another question (see section 3.4.5.10).

From Popper we have learnt that there is no absolute verified truth in scientific theories. The same applies to patterns. There are no “verified patterns”, only patterns with different degrees of corroboration based on the available evidence.

3.4.3.3 Abduction / Retroduction

Abduction is to look for a pattern in a phenomenon and suggest a hypothesis. Unlike deduction and induction, abduction is a type of critical thinking rather than symbolic logic. The objective of abduction is to determine which hypothesis or proposition to test, not which one to adopt or assert. Abductive reasoning usually takes the following structure:

X is observed.
Among hypotheses A, B, and C, A is capable of explaining X.
A is a probable explanation for X.

Simon argues against the “mystical view towards discovery” shared by creative scientists and artists (Simon, 1973). While the falsification method is a strong argument for testing a hypothesis it is not for discovering them. Referring to Hanson’s “logic of retroduction” (Hanson, 1958), he discusses an example of finding patterns in given data. His simple example is the following sequence of letters:

“ABMCDMEFMGHMIJMKLMMNMOPMQRMSTMUVMWXMYZMABMC….

Looking on the sequence, we find that there is a pattern in it:
$n(\alpha)n(\alpha)s(\beta)$; $\alpha = Z$, $\beta = M$

“where ‘$n(\alpha)$’ means replacing a symbol by the symbol next to it on the alphabet, α; ‘$s(\beta)$’ means repeating the same symbol as β; while the expression ‘$\alpha = Z$’ and ‘$\beta = M$’ set the initial values on the alphabets, at Z and M, respectively.”

Simon points out that we can be certain (i.e. verify) that it is a law for the given sequence. However, we cannot be sure whether it is an appropriate law for continuing the sequence (for the uniformity of nature cannot be verified): “…whether the pattern will continue to hold for new data that are observed subsequently will be decided in the course of testing the law” (Simon, 1973).

Finding the law for the sequence, Simon suggests, is released from the problem of justifying induction, because one can consider the process of finding laws without claiming that the discovered law is the unique description, or the most parsimonious possible. *Testing* and *finding* regularities can follow different norms.

Finding laws or law proposals (hypotheses) can be done in an efficient or inefficient way. The inefficient way is what he calls the “British Museum Algorithm” to “honor the monkeys who were alleged to have used it to reproduce the volumes in the British Museum” (Simon, 1973). In other words: this process, the finding of theories “out-of-the-blue”, is randomly or based on trial-and-error at its best. The algorithm might produce the right law for re-creating the pattern in reasonable time for simple cases. However, if we use a heuristic search algorithm and apply strategies to find the pattern, we might be far better off. The heuristic search algorithm “extracts information from the sequence in order to generate directly an alternative that will work. The difference between the two algorithms is exactly parallel to the difference between solving an algebraic equation by trying possible solutions in some systematic order, and solving it by eliminating constants from the left side, next eliminating variables from the right side, and then dividing through by the coefficient of the variable” (Simon, 1973).

It turns out that this approach, which is also applied in the process of pattern mining, is an appropriate way for finding new hypotheses. To believe that the law is likely to hold true in future cases cannot be said by this method – it offers only a probable hypotheses that must be tested.

3.4.4 Methods for pattern mining

The mining of patterns is an attempt to find the regularities and generative rules of design forms: “In all these cases, no matter what method is used, the pattern is an attempt to discover some invariant features, which distinguishes good places from bad places with respect to some particular system of forces. [...] It is in the invariant behind the huge variety of forms which solve the problem. There are millions of particular solutions to any given problem; but it may be possible to find some one property which will be common to all these solutions. This is what a pattern tries to do" (Alexander, 1979, p. 260).

In *The Timeless Way of Building*, Christopher Alexander (1979) names the following ways to find patterns:

- Observation and analysis of good examples
- Analysis of bad examples and inference of good solution
- Inference by pure argument

All three methods can be used to find invariant features that discriminate good from bad designs. They refer to different approaches in finding theories:

Induction: The observation and analysis of existing cases is inductive inference.
Inductive-Deductive: Analysing the commonalities of bad examples is an inductive inference which is followed by deductive inference of a working solution – the lessons learnt.
Deductive: Inference of good solutions by pure argument only based on theoretical assumptions.

While the deductive approach was the rational component of "The Program" in *Notes on the synthesis of form* (Alexander, 1964), Alexander later sees deduction only appropriate for occasional cases. Patterns are usually inferred from experience and not deduced from theories. An example for a theoretical pattern is the first pattern Alexander et al. described in *A Pattern Language* (1977) on INDEPENDENT REGIONS. This pattern has never been applied and whether it works or not cannot be tested. In the pattern community, the inductive approach is the agreed paradigm. Patterns are derived from practical experience and not deduced from theories. Discovering a pattern is called pattern mining. This metaphor emphasises the analysis of existing design structures and the implicit knowledge of experts. The process of pattern mining reveals "nuggets of wisdoms" from the structure and form of artefacts and the decision making of their creators. To expose the invariant structure and discriminate it from the surface structure (non-essential features) is the main task of pattern mining.

Very often the mining process involves a mixture of methods and follows an iterative process of identifying, refining, criticising and cross-linking patterns (Retalis, Georgiakakis & Dimitriadis, 2006). Inductive and deductive methods often go hand in hand (Baggetun, Rusman & Poggi, 2004). First categories are inductively derived from the observered data. Thereafter deductive analysis can help the articulation as patterns (Schadewitz & Jachna, 2007). Deduction is needed to reason about the patterns – the recurrent patterns found by induction do not *explain* the relationships. It is important to point out, once again, that the generalization from single cases, the reasoning about causalities of working forms and the judgement between relevant and irrelevant features is *speculative*. The pattern itself is a theory. Or, as Brad Appleton (2000) puts it: "A pattern is where theory and practice meet to reinforce and complement one another, by showing that the structure it describes is useful, useable, and used". As such, patterns are theories of good practices. Typical methods for the inductive inference of patterns can be found in qualitative research (Hughes, Rodden, Rouncefield, & Viller, 2000) and comprise techniques such as observation and analysis, retrospectives, expert interviews, focus groups.

Kerth & Cunningham (1997) and DeLano (1998) both name the following methods:

Introspective approach / individual contribution: self observation and analysis of one's own projects, which processes and designs have been successful or not. Since this approach explains the results rather than the internal feelings and thoughts of a designer, we suggest calling this the **retrospective approach** rather than an introspective approach.

Social approach / secondary contribution: Observation of the environment and the behaviour of its agents, interviews with experts who explain their own experiences or patterns. A methodology to extract patterns from case studies has been developed by Mor & Winters (2007).

Kerth & Cunningham (1997) mention additionally:
Artifactual approach: Observation and analysis of project results. Many of Alexander's architectural patterns have been induced by this method: by studying existing buildings. Many software design patterns are developed using this approach (Buschmann, Henney & Schmidt, 2007) by contrasting and comparing similar systems (Rising, 1998).

DeLano (1998) mentions additionally:
Pattern Mining-Workshops: Focus groups are used to collect, categorize and summarize the experience of experts. Such discussions often show that there are different views on the same issues which can be harmonized in the course of discussion.

Since pattern mining is a social process many methods are based on workshops. Iacob (2011) describes design workshops in which “participants are encouraged to externalize any design problem, solution, or decision they consider relevant to their design process”. Iba & Isaku (2012) describe a workshop format for a holistic pattern mining approach. Instead of mining single patterns the workshop aims at mining whole languages. Experiences are collected on sticky note and clustered visually by a group of participants. This process unreveals patterns and connections between them for a complete domain. Hanmer (2012) has written patterns for pattern mining that reflect on identifying gaps in a language and to distribute the writing in a group of authors. Winters & Mor (2009) desribe a process that involves public workshops to involve practioners in order to verify identified patterns and to generate new patterns based on design narratives (Mor, 2010). Their approach follows a structured process of naturalistic generalization.

All of the methodologies mentioned afore use inductive inference to gain new findings from the field. This is typical for qualitative research (Flick, 1998). The shepherding process (Harrison, 2006a) and the Writer’s Workshops (Gabriel, 2008a), too, are qualitative methods that primarily ensure the quality of the pattern description. Both methods help to find errors, gaps, and ambiguousness in the description. Usually the shepherd and workshop participants are also familiar with the subject and can sometimes support the pattern with additional cases and variations of the pattern. Hence, the pattern writing process is also a method of pattern mining because it reveals new facts either by adding additional experiences or asking the authors to express more of their implicit knowledge in the written pattern. Since the patterns are presented to other experts of the field, the workshop is a first test for the patterns because workshop participants can oppose to the content of the pattern description.

At EuroPLoP 2009 we used an open space slot to collect different approaches of pattern mining. The workshop confirmed the methodologies presented in this section. Quotes that referred to inductive methods include: “watching out for keywords when talking to people (…’that’s always like that…’)”, “find out about that worked”, “talk to practitioners and domain experts, collect success stories and case studies”, “reflection on own experience and discussing about it with others”, “validation through others”, “abstraction of common features of products”, “from concepts”, “from products”, “from applications”, “analyze a number of artefacts, look for good stuff in it (recur. forms)”, “reverse engineering”, “group discussion”, “work in a team of 5-12 to find patterns and brainstorm issues, work in team of 2 to write”, “A group session discussing a problem that comes up with a solution”. Other quotes point to deductive approaches: “filling gaps in pattern language”, “from standards and regulations”, “splitting a complex solution into simpler individual patterns”, “narrative consequences on one pattern often lead to other patterns”, “digging deeper into a problem and finding the ‘laws’ which are behind”, “analyze literature for patterns”, “look for ‘patterns’ in non-pattern literature and make them pattern literature, e.g. fill the gaps”. Most participants mentioned different ways of reflection, generalization and abstraction. Very often a combination of methods is used and individual contributions (single or small groups of authors) are combined with second order contributions (feedback, discussion with others). In practice most of the pattern mining processes follow the procedures and techniques of grounded theory (Strauss & Corbin, 1990).

There is also research on automated pattern mining. Dong, Zhao & Peng (2009) provide a review of different methods and tools. The trouble with automated pattern recognition is that tools can hardly identify meaning. The methods presented in this section suggest that pattern mining rather is a social process.

3.4.5 Confidence and corroboration

The previous chapter has shown that there are various methods for pattern mining. The pool of qualitative research methods may offer additional ways to mine patterns. Unfortunately, not every pattern paper reports about which methods have been applied and in which environments the cases (the pattern’s substance) have been found. However, such information is critical to judge the confidence and scope of a pattern.

Alexander has used asterisks to indicate his level of confidence in his patterns and some pattern authors have adopted this style. But this is a rating done by the authors and likely to be biased. In particular, the scope of a pattern’s context may be limited to the experiences and attitudes of the authors. Information about the applied mining methods and the mining field could support less biased confidence. The sections in this chapter will discuss which meta-information can supply less biased indicators for confidence and corroboration.

3.4.5.1 Levels of objectivity

Both the mining ground and the chosen method have implication for the objectivity of a pattern. There is a difference between a single author reporting her/his patterns and the outcomes of a group discussion. Likewise the range of domains and contexts in which a pattern has been observed is critical to its general applicability. For example, if a pattern has been observed multiple times in Java programs, does this necessarily imply that it will work for C++ code as well? Without having observed or tested it, one cannot really (empirically) tell.

Since the creation of categories depends on the objects known and the properties considered, we can hardly speak of objective categories. Note that objective does *not* mean closer to the truth but that judgments (deciding whether one configuration is of a specific category or not) are inter-subjective. There may be objective laws how people perceive whole forms and construct their mental categories and patterns, e.g. the gestalt laws (Wertheimer, 1938; Goldstein, 2009), Alexander's 15 fundamental properties (Alexander, 2000a) or schema theory (Kohls & Scheiter, 2008). While in these cases the process may be universal, the results are not. The traces of perception, experience and manipulation of mental structures are a unique history for each individual and therefore result in personal views of the world.

If some objects do not vary very much, e.g. soda bottles, it is easier to denote a category people agree on. If there are only some commonalities it gets harder, e.g. a lecture can have many forms. People not only have different ideas about what an ideal lecture looks like, they might even debate if an extraordinary lecture is a lecture at all. The functional forces documented in a pattern are even less objective because different individuals may care for different functions and values. Which functions are critical and relevant depends on the interests of the individual. The least objective properties are those of beauty. Of course, there are objects most people consider as beautiful, e.g. rainbows or water lilies – but just as well, they might consider a picture of these objects as mere kitsch. In the aesthetics of Kant, Hegel, Wittgenstein and others (Majetschak, 2007), beauty and wholeness depends on our familiarity with a category. Somebody who is competent in the domain is capable of giving reasons for her/his aesthetical judgement. For example, we cannot only say that an OBSERVER is a beautiful design (in specific contexts) but also give reasons why it is the right thing. It is not just a solution but a good solution. A beautiful solution is one that does the right thing and fits well within our own aesthetic categories. In the case of the OBSERVER, being a programmer is necessary to see the inherent beauty of this pattern. As a programmer one can see the beauty in good program code (Oram & Wilson, 2007).

Individual categories and normative categories (e.g. ethical values) are counter arguments for objective and true categories, because such categories can change and have changed over time. Many norms in software design appear to be reasonable conventions, e.g. maintainability, robustness, performance, memory optimizing, etc. These conventions are implicitly agreed upon in the community of practice. Teaching practice, apprenticeship and also software design patterns make these categories explicit, and thereby inter-subjective agreement becomes more likely.

3.4.5.2 Justification for induction

Individual skills, attitudes, beliefs, volition and prior knowledge are as much part of the context of a pattern as any other environmental aspect. People may prefer different styles of architecture, painting, or coding. What people prefer and value has changed over time and often depends not only on regional cultures but on peer-groups and communities. Hence, the cultural setting and background is critical for the contextual validity of a pattern.

This calls for more transparency in pattern mining. To evaluate a pattern, it is crucial to learn in which environment a pattern was mined, which attitudes and beliefs the author held, and how many different views and artefacts have been examined to come up with the pattern. To assume that one designed form will work in other cases and for other people very much depends on how stable the past cases have been and how similar the new cases and cultural settings are.

Again, we can learn from the theory of science, since here we find established factors which are critical for the justification of induction. Westermann & Gerjets (1994) name four factors that influence the justification for inductive inference to new cases: sample size, validity and variation of previous positive cases and the similarity to a new case.

Transferring these justifications to the realm of patterns we can say that a pattern is better justified if there are **more positive examples** it builds on. Also, the number of negative examples (cases where the claims of the pattern are false) should be low compared to the positive examples. The **validity of the cases** depends on the objective perception of whole forms (which has its own problems as the last section has shown). Another justification comes from the **variation of cases**: if there are a lot of different cases in which the pattern succeeded, it is more likely to be a justified plan for new designs. For instance, a pattern that claims to have a wide scope of application is better justified if it has actually worked in different settings and for different domains. The pattern is also strengthened if the negative cases are similar (e.g. the pattern does not fail generally but under certain conditions – such conditions could change the described context in the evolution of the pattern). The justification for a designer to choose a particular pattern for her/his design is better if the current design problem is **similar to the positive cases**, and it is lower if it is similar to negative cases. This implies that the designer has to share the same intents, values and attitudes the pattern is based on.

3.4.5.3 Empirical and logical content

The extent to which we can test a pattern depends on its empirical content. The confidence we can put in a pattern does not only depend on the number of cases it is based on but also on how much a pattern claims. If a pattern claims to work in many different cases (broad context) it must be testable in many different cases. If a pattern gets very detailed and specific in its solution it claims more than a pattern that lets unanswered a lot of questions. The level of preciseness of a pattern measures its content, i.e. what is contained in a pattern.

We can distinguish between the logical and empirical content of theories or patterns respectively. The two compounds are opponents: A theory with a small amount of logical content has a high amount of empirical content. A form (situation or phenomena) that is fully specified logically contains exactly one case. On the other hand, forms that are not specified at all contain any kind of configuration - the logical content is maximized because logically all cases are contained. Let us consider the logical and empirical contents for patterns.

3.4.5.4 The narrower a context is, the lower the empirical content is

If there are a lot of conditions conjunct in the context, that means that it narrows the number of cases for which it makes definite statements. For all cases not included in the context we do not know whether or not the pattern works, hence for such cases we have logically two different outcomes (applicable or not applicable). If there are only a few cases for which the pattern claims to be applicable (This pattern can be used if a AND b AND c…) this leaves only a few cases for which we can test the pattern. The context is very narrow.

As an example consider a pattern which states in its context that it can be used for the programming language Java and the domain of banking. That it can be used for banking applications does not say that it cannot be used for other domains as well. It just does not make any claims about it. Let us only consider what is 1.logically and 2.empirically contained in that pattern:

a) The pattern can be used for banking domains AND
b-1) the pattern can be used for game programming OR
b-2) the pattern cannot be used for game programming.

Only a) can be tested empirically because b) is a tautology: it either works or not in the context of game programming. Logically we have three cases because neither b-1) nor b-2) is excluded.

If we extend the context and claim that the pattern works for both banking domains and game programming it contains:
a) The pattern can be used for banking domains AND
b-1) the pattern can be used for game programming domains.

We can now test the pattern in two domains, banking and games. Hence, the empirical content has grown. It claims more and there is more that can be tested. The logical content has shrunken because we have explicitly excluded b-2).

3.4.5.5 The broader a context is, the higher the empirical content is

If there are a lot of alternatives in the context (e.g. a pattern works likewise under these conditions AND those conditions), then the pattern claims a lot. For example, if a pattern says that it can be used for banking AND game programming AND network environments, then we have more cases in which we can actually test it. Similar to the previous example, by saying that a pattern works for network environments as well, we have eliminated the tautology "does or does not work for network environments" in favour for a decision. Note that the tautology could also be eliminated by claiming the opposite, e.g. to state in the context that the pattern is not suitable for network environments.

Generally, each situation that is not explicitly included or excluded in the context description remains unspecified – that is, the pattern makes no claims about whether or not it works in such cases. If $\{c_1, c_2, \dots c_n\}$ is the set of all possible contexts 1..n, then a context description that states only c_1 says nothing about $c_2, \ldots c_n$. To say a pattern works in all contexts is a shortcut for saying it works in contexts $c_1, c_2, \ldots c_n$. To say a pattern works only (exclusively) in context c_1 is to say that it works in c_1 and does not work in $c_2, \ldots c_n$.

A word of caution at this point: These examples are to show how statements affect the empirical and logical content of a pattern. This is not a call to actually use such formal statements in the written patterns! It is only to make the reader aware in which way the contexts "banking domain", "banking domain and game programming", "all domains", or "only in banking domain, no others" differ in their claims. A pattern should only claim as much as has been tested, e.g. to say it works "only in banking domains" when there is actually no data whether or not it works in other domains is as bad as claiming that a pattern works in all contexts when there are some in which it does not.

3.4.5.6 The more precise a solution is, the higher the empirical content is

Likewise, the logical and empirical content of the solution depend on each other. If we precisely describe what design options are allowed, should be avoided, or have to be done, then we are explicitly limiting the number of possible solutions. The more constrained a design space is, the fewer designs are contained in it – therefore its logical content is low. Because of the precise and detailed description, the empirical content is higher. There are more things expressed about the design and more things can be tested (and falsified) accordingly.

Consider this example: If you have a pattern for a method in online education that tells you one possible time span of running it, the empirical content is low. For example if you say that an ONLINE TRAINING (see appendix F) can adequately last 30 minutes then you have exactly one time value to test. Its empirical content is low because it offers only one test case. Its logical content is high because it does not claim anything about trainings that last 29 or 31 minutes. Hence, for a training that lasts 31 minutes it logically includes both cases, the one in which the time span works and the one in which it does not. If you extend the claim in the solution statement and say that a time span between 30-45 minutes is adequate, you increase the empirical content. You make the solution more precise by saying that 30 AND 31 AND 32 … AND 45 minutes will work. Hence, you logically exclude the case that 31 minutes are *inadequate* for a training claiming that it is always *adequate*. By giving a range of possible values we extend the assumption. If it is true that all the options are equally valid solutions then this is a desirable extension. We can also extend the empirical content by explicitly saying that a training of 29 or 46 minutes will never work because such claim can be tested: running a successful online training of 46 minutes would falsify the assumption that 46 minutes are inappropriate[22].

3.4.5.7 The less precise a solution is, the lower the empirical content is

If the solution is less specific and suggests different paths without saying which one will actually work, its logical content is higher because it contains alternative forms. But it does not tell you which of the alternatives will actually work. To say that either A *OR* B will work logically means that it is left open which one actually does (to say that both forms work would logically be expressed as A *AND* B work, thus, the statement would be more specific not less).

To understand the implication, imagine that you have a specific design problem and somebody tells you that **one** of the Gang of Four or POSA design patterns will help. The logical content of this solution proposition is high since all the patterns are included. The empirical content is low because it does not tell you which one will work.

A more extreme example is Joseph Bergin's pattern "Do the right thing" which makes fun of the idea of having a universal pattern. Its solution is very general and unspecific: "Do the right thing. Make the bad thing better." Its logical content is maximized. Doing the right thing could be anything. In fact, if you consider *anything*, then *something* will work. Therefore its empirical content is zero. We know in advance that because of the logical structure the statement is always true. It is a tautology! There simply is no way to falsify this pattern because it includes all possible things to do that *might* make things better. The point is, it does not tell us **which** of the many things to do.

[22] Of course, the exact bounds for adequate time spans of online trainings are fuzzy and not precise. Bean counting the minutes was only to demonstrate how the logical and empirical content change if the range of specified values is modified.

3.4.5.8 Testing the empirical hypotheses

So far we have only talked about context and solution. Since we have seen that a pattern is not just a simple hypothesis (*if* context *then* solution) but a network of hypotheses that explain the forces that cause the fitness between context and solution, these hypotheses count for the content as well. Each force tells something about the context and problems, and each force can be falsified empirically. That is, we can test whether all the forces actually exist in a given context and whether all the forces are actually resolved by a given solution.

The volume of empirical content tells us how much there is to test and how much we can learn from a pattern. However, it does not tell us how often the pattern has actually been tested. The more a pattern claims (a broader context or very detailed statements about what works and what does not) the more tests have to be performed.

It is important to notice that we can test all the claims and forces empirically, but we cannot do this in isolation. We cannot test a single force or a single design variable ceteris paribus. The reason is that there are interdependencies between the form variables. This is typical for complex systems "that just do not allow the varying of only one factor at a time – they are so dynamic and interconnected that the alteration of one factor immediately acts as cause to evoke alterations in to others, perhaps in a great many others" (Ashby, 1956, p. 5). If one puts a different weight on a single force (e.g. change a force ceteris paribus), the complete design may have to change. We can test a pattern only as a whole. As a consequence each test must include all the statements that are contained in a pattern. Usually this includes a lot of implicit tests and the amount of empirical content is indeed critical to justify the "Rule of Three" as statistically significant evidence for prior positive cases.

This section has illustrated how the empirical content changes. For clarification we have used very simple statements of which we have assumed that they can be considered in isolation and that they are discrete. By all means, this was only for illustration. We should not start to write more formal patterns with isolated statements. Not only would this contradict the usability for the target audience. It would fragment the pattern into isolated parts reducing it to mechanistic analysis and contradict the very idea of Christopher Alexander.

3.4.5.9 The Rule of Three

The "Rule of Three" informally suggests that there should be at least three known uses. A singular solution is just a design, two occurrences might be coincidence, whereas the third occurrence makes a solution a "pattern"[23]. Of course, the recurrence of a design configuration could still be random. Then, why do we accept the "Rule of Three" heuristically as sufficient evidence to talk of a pattern? The question that we have to ask is: how likely is the invariance of successful design configurations just random outcome? The answer is that the statistical significance depends on the logical and empirical content of a pattern. Let us assume that we have a number of different design objects codified in binary form by representing various design variables and relations as binary strings. An example is shown below:

001101010001000100001001   0010101010111111101011101   110100001010100011111001
111100011111011110001001   110100001010110011101011   100001011010111010100001
110100001010111110001110   111101100001111100010001   111111011101101110001111

When we are looking for patterns in the design of objects, we usually look for recurrences[24]. The pattern "1001" appears three times[25] at the end of a string, the pattern "1101000010101" appears 3 times at beginning. How likely is it that these recurrences are just random? For the "1001" it is quite likely because if you take only four binary variables into account, roughly every eighth (2^4 =8) object could have this configuration by chance. Its logical content is huge because there are many objects in the world having randomly this configuration. Its empirical content is low, because there is not much we can test and learn from. The other pattern, however, is a string of 14 digits. Thus, it should recur by chance only one out of 2^{14}=5096 times[26]. In our example it appears three out of nine times. This could still be a recurrence by chance, but it is less likely. The "Rule of Three" is significant in most cases because design patterns usually have a high amount of empirical content in their solution parts. Inseparable design variables suggest that a single test consists of multiple fallible statements. It is

[23] http://c2.com/cgi/wiki?RuleOfThree
[24] More precisely we are looking for whole forms or perceivable coherent gestalts that recur.
[25] To simplify, we do not distinguish between context and solution. Rather, we only consider the solution forms and assume that all objects are satisfying solutions to the same design problem.
[26] Note that the chance to find *any* pattern in a huge number of objects is much higher. This is similar to the birthday game in which you need only roughly 20 people in order to find two persons with the same date of birth by 50% chance. Still, the reoccurrence of *complex* patterns is much less likely by chance than of *simple* patterns. See also appendix C.

just not very likely that in design objects tackling the same problem similar configurations occur again and again by chance.

This is not an advocacy for finding absolute truth through inductive reasoning. We are in line with the critical view Hume has put forward in the middle of the 18th century in his work “An enquiry concerning human understanding”[27]. If we take any regularity as a causal pattern, claiming a causal connection, we may end up like Aristotle. He noticed that mice were commonly found in barns where grain was stored. He thought that the mice grew from the grain and hay, and he coined the term “spontaneous generation”, the hypothesis that living organisms arise from nonliving matter. As a matter of fact, he published a recipe that anyone could use to grow their own mice: darkness + hay + grain = mice.

The “Rule of Three” can make a pattern significant but not necessarily plausible or true. But this is a problem shared by any statistical method that can only capture correlations. As such the “Rule of Three” is no better or worse. The difference is that due to the complexity of statements tested at once, fewer cases are needed to make the invariance significant. Again, the binary representation was just for illustration: real design problems are far more complex and binary representations might be possible in principle but not practically.

3.4.5.10 Testing a hypothetical pattern

As a matter of fact, if we consider a limited set of objects (as in the previous example) it is not even likely that a specific pattern (logically only a few configurations are allowed) re-occurs two times by chance. So, actually, three recurrences is the minimum number of cases required to test an induced pattern (from two occurrences) at least once. Hence, the “Rule of Three” sometimes implicitly includes a test of a pattern. For example, some of the patterns for interactive information graphics (Kohls & Windbrake, 2006, 2007, 2008a, 2008b) were mined using the artifactual method by investigating various multimedia applications (Kohls & Uttecht, 2009). A first recurrence of interaction forms qualified the form as a pattern candidate. Another occurrence qualified it as a pattern. In this case, the pattern candidate was the hypotheses (an assumed invariant form) and another occurrence was a first corroboration. This works also with other mining methods such as individual contributions: Very often, we draw from our experiences and think “this could be a pattern” (we have a hypothesis). Then we see the assumed pattern applied in another design and think that it is indeed a stable pattern. We have our first qualitative corroboration.

Of course, it is desirable to have patterns and the respective pattern descriptions tested more systematically. Pattern descriptions usually include a section entitled “known uses”. However, we never learn whether the pattern was induced from these cases or whether (some) of the cases have been used to test or support the pattern.

Furthermore, it is of interest to evaluate the quality of a pattern, whether there are successful uses of the pattern description, i.e. its actual application to new cases by third parties. It would be beneficial to learn about failures as well. Unfortunately, authors usually do not learn about the application of their patterns. As a first step, pattern authors should encourage readers to provide feedback about the application of their patterns. Summaries of the feedback – positive and negative – should be part of the pattern description, provide evidence or counter-evidence for the quality of a pattern.

3.4.6. Knowledge captured in patterns

To find a working relation of form and context – a successful solution – means significant scientific progress. We learn much more from best or good practice than we learn from failures. If there are a possible million ways in a landscape, the number of ways that lead to the goal are relative small. To know that way_1 works, and way_2 does not work, contains different amounts of knowledge. If there are 100 ways that work and a million that do not work, we can actually calculate the information content that we get if we have tested one way. There are fewer micro-states for the macro-state “this form works” than there are micro-states for the macro-state “this form fails” (see section 3.2.8.1). Hence, we are better informed by the macro-state “this form works” than “this form fails”. Our knowledge is maximal – i.e. complete – if we know of all ways that work and which do not work. We can get this knowledge by listing one of the two sets: all ways that work or all ways that do not work. The smaller set – the working ensemble – is more efficient to list. If there is linearity, instead of listing the single ways that work, we could define the boundaries of the space in which we find the ways that work.

[27]Online edition: http://18th.eserver.org/hume-enquiry.html

3.4.6.1 Boundaries and information content

The distances between two boundaries may vary at different places. Moreover, most designs are multidimensional and include a number of design variables. Nevertheless, for each design variable we are looking for ranges that work rather than single values that do or do not work. "A system is said to be stable along certain of its variables if those variables remain within defined limits" (Watzlawick, Bavelas & Jackson, 1967, p. 185). The ranges of valid variable configurations are not independent, i.e. one variable depends on others. A pattern describes a stable configuration of variables. But it does not describe single variable values but rather ranges of interrelated variable configurations. In that sense a pattern describes a multidimensional design space. Single designs are specific configurations of the design variables within the bounds span by the pattern.

By describing boundaries rather than single configurations that work or do not work, the information content and, thus, the knowledge captured in a pattern is much higher than the description of single best practices – or worse single failures. However, the configuration of best practices and failures provide hints to the range a pattern spans. Practically it is impossible to define the exact bounds of working configurations. For example, one can say that a seminar works very well with 8, 15, 25 students; a seminar with only 2 students or with 40 students may have failed. This provides hints for the optimal group size of a seminar. Its lower limit is somewhere between 3 to 7 students, its upper limit somewhere between 26 and 40 students. Over time, a more specific range may emerge, for example a seminar should have at least 6 and no more than 30 participants to work properly. These boundaries, however, are not absolute but rather provide probabilities for success. A group size of 5 students could work as well but it is less likely than having a group size of 12. Likewise, the optimal duration for an online training cannot be quantified exactly. If we say that an adequate duration is between 30-60 minutes, we imply that within this range a success is more probable. But 70 minutes training could still be interesting while a 45 minutes training is sometimes boring. It is just that there are more 45-minutes trainings that work than there are 70 minutes trainings.

3.4.6.2 Information content at different positions

Let us assume that we conduct a research involving 10.000 online trainings; after each training the trainer and the participants are asked whether the time was appropriate. A fictive outcome would show that for each time span there is a different probability of appropriateness. The most exact information would be included if we know the probability for each time span based on tests. However, if we just take the range of durations that work most of the time, we can simply say that the duration of 30-60 minutes is fine. This statement would cover most of the information because we can predict the outcome most of the times. We can say that we are informed that a 45 minutes training will work. If it does not work we are more surprised. Likewise, we are informed that a 90 minutes online training will probably fail. If it does work nevertheless we are more surprised.

That shows that there is implicitly an underlying information structure that lets us expect a certain outcome for each time span. Of course, we are rarely in the position to conduct a research of 10.000 online trainings. Moreover, the appropriateness of time depends on many other factors such as content, the trainer, and interest of the students. The point to make was that at the boundaries, the information content is highest: 65 minutes could be either too long or not – we only know after the training. 90 minutes, however, are most certainly too long; 45 minutes are most certainly still appropriate.

3.4.6.3 Approaching the boundaries

In practice, we do not need to observe 10.000 online trainings with different time spans because once we know that 90 minutes will not work, we do not have to test whether 100 minutes work. Rather, the approximate zone of boundaries emerges after a couple of runs, e.g. 45 minutes (good), 50 minutes (good), 80 minutes (bad), 55 minutes (good), 70 minutes (OK), 55 minutes (good), 60 minutes (good), 65 minutes (bad),….

This sequence shows that we are learning about a boundary by approaching it by testing different values and adapting our assumed limit. After just eight runs we know that up to 60 minutes the outcome is always good but beyond that we get only OK or even bad outcomes. Boundaries are revealing much more knowledge than single failed or best practices. Of course, an expert usually does not measure the range of valid values (e.g. time spans for an online training) in experiments but builds tacit knowledge as rules of thumbs based on experience.

3.4.7 Implications for design patterns: Is it a science or an art?

If science is about the nature of things, then patterns certainly belong to science. In the case of patterns, the things or objects of consideration are artefacts and practices of creating the artefacts. Hence, patterns are a way to investigate the "science of the artificial" (a term coined by Simon, 1996), or the nature of artificial objects. However, patterns are not about science only. The scientific component of the pattern approach is the mining of invariants in both designs and the design processes, and to investigate the reasons and causes for specific forms (what forces a form to take its shape to serve in a specific context). But to actually run the process and to create the artefact is a matter of skill and craftsmanship, and this is where the pattern approach has its artistic component. That there is a difference between the general rule and the application of a rule is pointed out by Aristotle already (Schnädelbach, 2003). Knowing the rules of painting does not imply that you can actually paint. Tacit knowledge is always richer than any explication of that knowledge. There are things that cannot be communicated appropriately because such know-how is not only deeply in the individual but may actually depend on the individual (Polanyi, 1958; Mitchel, 2006). Explicit rules are helpful to communicate good practice but it is not said that such rules actually exist internally. When we perform or solve a design problem we are not aware of our thinking processes. The reasoning comes afterwards and has the same structure as the explicit rules that we can communicate. This, however, is not a proof that we have internally such a rule-based knowledge if we take Polanyi's idea of tacit knowledge seriously (Neuweg, 2001).

A pattern is neither a necessary nor a sufficient condition for successful problem solving. One can solve problems without prior knowledge of patterns. And even the best pattern will not help if an individual fails to understand and internalize its meaning and successful practical application. The reliability of a pattern is a necessary condition to be useful but it is not sufficient. The quality of a pattern highly depends on its usefulness, i.e. the complexity and relevance of the problem solved and the accessibility of the pattern description.

Still such a pattern could be pure fiction. We have several times referred to Newton's laws. While these "laws" are useful approximations, according to the theory or relativity the laws of physics are different all together. However, you can observe the differences only in extreme situations – Newton's laws work in most but not in all contexts. Therefore, they are very useful without being true. Likewise a pattern can be very useful without being true. Maybe the notion that every pattern tells a story should be taken more literally. But from a pragmatic point of view what matters is not the truth but the usefulness of the patterns in solving important problems.

We hope to have shown that patterns are theories and not "real stuff" but rather "real theories". They are inductively inferred from "real stuff". While we have argued that induction and abduction are appropriate methods to derive hypotheses, we must stress that this does only explain the designs of the past and that alternative explanations can always be given. Therefore, the hypothetical explanations and design predictions have to be tested in order to strengthen a pattern. In fact, a good designer tests a pattern before s/he uses it, e.g. decides whether a pattern actually fits the situation at hand. Patterns do not release the designer from the responsibility to think about the design. For the pattern community, it is important to realize that each application example only provides evidence and not proof that a pattern actually works. Proofs can only be provided for pure mathematical inference and logical deduction.

As an outlook, we suggest that pattern papers should not only include the pattern descriptions but also give more space to document:

- The mining ground (variation of cases)
- The mining methods (validity of cases, confidence, objectivity)
- The known uses that induced a pattern (sample size and variation of cases)
- The degree of corroboration (successful application of patterns to similar cases as evidence)

To be sceptical about each individual pattern does not weaken but strengthen it. If a pattern can fail in principle but shows to succeed continuously, this evidence makes it more reliable.

4 Empirical studies on patterns

4.1 Introduction

The previous chapter has described the foundation for a theoretical framework that distinguishes between the patterns in the world, the patterns in our heads and the explicated pattern descriptions. The three main sections of this chapter are the symmetrical counterparts to chapter 3 and should support the theoretical framework. Each of the three pattern forms is related to other research disciplines and therefore we need to use different methods to approach the research questions. The generation of patterns in the world is tackled by research methods that are common in computer science: a case study of a concept implementation and the lessons learnt. The construction of patterns in our head is subject to psychological research and we will follow common experimental setups to test hypotheses and report the statistical evaluation. The epistemological questions about the information content in pattern descriptions are tackled by qualitative methods. We will analyze different pattern descriptions in order to compare the available statements that can be tested.

4.1.1 Pattern mining and generation of interactive graphics

Section 4.2 contains a case study about the implementation of a multimedia authoring system that generates pattern instances of interactive graphics. We will first discuss the functions of that software and how users can define new interactive graphics using wizards and a visual language. The set of available pattern generators is the result of a pattern mining process using existing interactive graphics. While the induced patterns may or may not be appropriate, as section 4.3 will discuss, the programmatic specifications are objective and universal. That is the same code fragments generate replicable interactive behaviours in the runtime environment. By specified variations the wizards generate an abundance of different artefacts based on the same patterned rules. The output might look very different based on the parameters but the algorithms are universal. However, the selection of interaction patterns and their splitting into different behaviours is only one of many alternatives. Whether the division and abstraction is reasonable and shared in the world view of individual users is another question and subject to psychological research in the subsequent section. The general lessons learnt from a system that generates recurrent yet varying structure is summarized at the end of section 4.2.

4.1.2 Schema induction of similar interactive graphics

The next section corresponds to the patterns in our heads as outlined in the theoretical framework. We have claimed that individuals construct patterns based on their experiences according to schema theory. In this section we will test whether individuals develop stable mental patterns over time, whether they develop similar patterns, and whether some patterns are more coherent than others. Although we use our own patterns we can expect valid results from these experiments because the individuals do not know the underlying patterns in advance. Based on selected patterns we will create a sample of 14 interactive graphics that are used in pair comparisons, classifications and description activities. We will first test whether instances of the same patterns are perceived as similar by the participants. We also expect that over time the judgement of similarity becomes more correct because individuals construct the patterns in their heads. The classification activities will show whether individuals have found the same patterns or classes of interactive graphics. We will identify the most frequent found classes as canonical classes and compare them to the expected patterns. A cluster analysis can show whether the hierarchy of mentally constructed patterns is symmetric to the hierarchy that was used in the multimedia authoring system. While the classification shows which graphics have been perceived as similar it does not show whether participants have recognized the general structure of interactions. To test this all participants have described the interaction of the class they have felt most confident about. We expect that the canonical (most frequently identified) classes are more coherent and descriptions of these classes should be more precise.

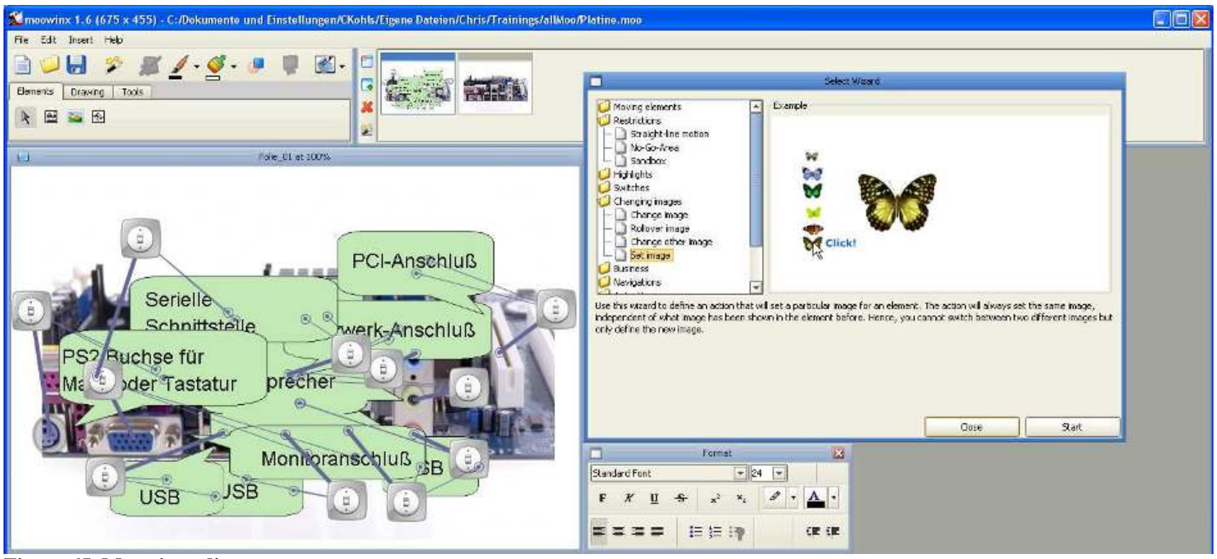
4.1.3 Analysis of pattern descriptions

In this part we will analyze different descriptions of the same patterns and compare pattern collections to pattern languages. This retrospective approach will show that the description format has impact on the information content. It supports the view that patterns can be treated as theories and may have different sets of statements that can be tested. We will see that the pattern descriptions are not the *Stoff* itself but only theoretical constructs that can miss important information (in the process of abstraction), provide wrong information (statements that can be falsified) and represent information inefficiently (overlapping patterns lead to information redundancy).

While this section should mainly support our claim about the theoretical nature of pattern descriptions it also illustrates how the efficiency of patterns depends on their granularity, abstraction and the mental ideas of the authors.

4.2 Pattern mining and generation of interactive graphics

Moowinx is an authoring tool to generate interactive graphics, mainly for educational purposes. In the current version of moowinx, building blocks can be used to define the interactions among visual objects such as drawings, shapes, input fields or images. These building blocks generate behaviour of the objects at runtime. Since the same behaviour of objects recurs over and over, these behaviours are patterns of interactions. To setup the behaviour of objects, moowinx offers a set of wizards that generate behaviour configurations. These wizards, too, contribute to the generation of interaction patterns.



Use this wizard to define an action that will set a particular image for an element. The action will always set the same image, independent of what image has been shown in the element before. Hence, you cannot switch between two different images but only define the new image.

Figure 65. Moowinx editor.

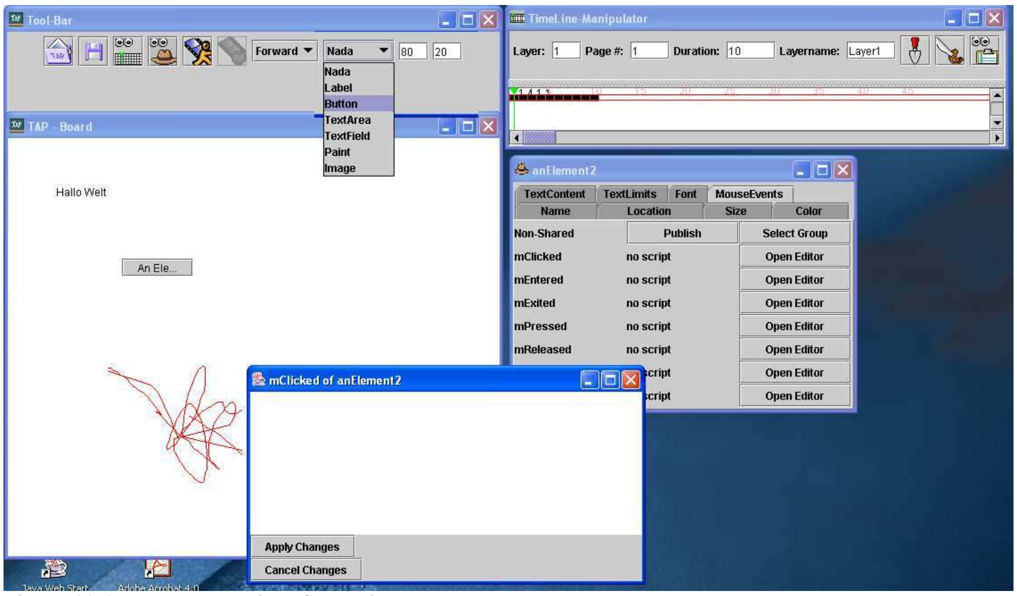
Interactive graphics could be used as learning objects in a self-paced online course or in an on-site presentation. It allows dynamic visualizations and interacting with slides. The later scenario is particularly interesting when combined with the use of interactive whiteboards. Presenters or learners can use their fingers to move and manipulate the objects on a large interactive projection. Interactive visualization helps people to analyze, understand and communicate models, concepts and data (Schumann & Müller, 2005). It is used for exploration, monitoring, confirmation and presentation of information. According to the Multimedia Principle (Moreno & Mayer, 1999; Mayer, 2001) there is evidence that the use of multimedia can improve learning. Dynamic and interactive pictures can be even more effective (Rieber, 1990; Rieber & Kini, 1991). Causal relationships can be directly and unequivocally perceived (Ware, 2004). Sequences of object or data movements can be captured. Processes of restructuring or rearrangements can be made explicit. Complex spatial actions can be shown. Cognitive load can be reduced by animations, too, because the changes over time are directly shown.

Moowinx offers a simple way for non-programmers to define responses of visual elements to the direct input of users.

4.2.1 Development of moowinx

4.2.1.1 Early versions

Moowinx started in 2000 as a student project in a laboratory course for software development and design at the University of Applied Sciences in Wedel (Kohls & Windbrake, 2003). The tool was first named TAP because our original idea was to develop an authoring environment for teaching and presenting (tap). These initials still occur in many of the class names of the software architecture. The tool was later re-named to jtap, Javanti, ActiveSlide, and finally "moowinx". The name changes were necessary due to conflicts with names of other projects or products. Along with the name changes the software developed at several stages. The first stable version was the outcome of my diploma thesis (2001). It included a timeline to organize interactive objects in slides on several layers, key frame animation, and a visual editor to arrange visual objects such as text labels, buttons, input fields, or images on a slide. It also included the scripting language Tcl with extended commands for animating objects (Kaiser, Kohls & Windbrake, 2002).

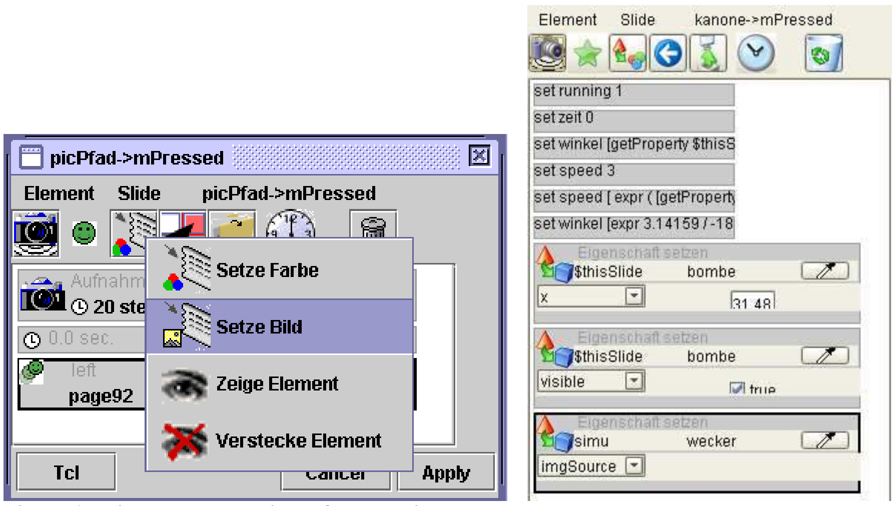


Non-Shared	Publish	Select Group
mClicked	no script	Open Editor
mEntered	no script	Open Editor
mExited	no script	Open Editor
mPressed	no script	Open Editor
mReleased	no script	Open Editor
	cript	Open Editor
	cript	Open Editor

Figure 66. A very early version of moowinx

4.2.1.2 Visual programming

In the years that followed, several students have developed new components for moowinx as part of their diploma theses, including 2D shapes, free hand drawing, and a component that allowed formatted text. For my master thesis (Kohls, 2004) I implemented a visual language for moowinx that allowed visual definitions of actions and behaviours of visual elements.



```
set running 1
set zeit 0
set winkel [getProperty $thisS
set speed 3
set speed [ expr ( [getPropert
set winkel [expr 3.14159 / -18
```

Figure 67. Visual representations of a Tcl script.

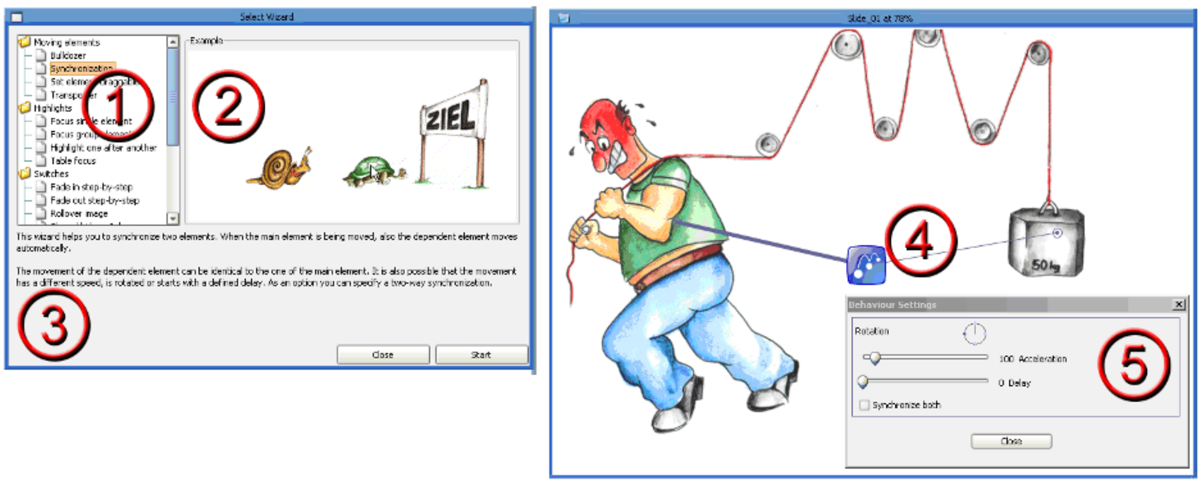
4.2.1.3 Patterns of interactive graphics

At that time, Tobias Windbrake and I were both working at the e-learning lab of the University of Applied Sciences in Wedel. Together we developed patterns of interaction for moowinx. We also specified wizards to generate these patterns; the wizards were implemented by Henning Heuer who worked for our company pharus53. Unfortunately, moowinx never became a commercial success. Fortunately, it was a perfect tool for the study of patterns since it exemplifies patterns of different kinds: patterns to create patterns (wizards), patterns to create interactions of elements (behaviours), software patterns to implement the behaviours, and e-learning patterns to discuss the pedagogical meaning of an interaction type. All of these patterns are designed patterns but not all of them are described as design patterns. We have seen that there are several levels to both the discussion of a problem and its potential solution (see section 2.3.2.3). An e-learning pattern addresses a pedagogical problem (such as limited cognitive attention), suggests a solution (such as highlighting objects temporarily), and discusses appropriate contexts. The ON/OFF SWITCH is an example of a pattern that defines an interaction behaviour that addresses this problem (see appendix E). A "behaviour" in moowinx is the manifestation of the interaction form described in the solution section of the design pattern. It solves the problem of implementation.

4.2.1.4 Wizards for patterns

A wizard that creates and configures the behaviour is a solution to a different problem, namely end-users may not know how to set-up the behaviour. For the same pattern of interaction, one can think of different sequences (wizard steps) to set-up the configuration.

The programming behind the interaction is yet another problem – a problem of code implementation. There are different ways to achieve the same emergent interactive behaviour. For example, one can create the same interaction in a Flash movie or a Java applet, however, the implementations are very different and involve different degrees of effort.



This wizard helps you to synchronize two elements. When the main element is being moved, also the dependent element moves automatically.

The movement of the dependent element can be identical to the one of the main element. It is also possible that the movement has a different speed, is rotated or starts with a defined delay. As an option you can specify a two-way synchronization.

100 Acceleration
0 Delay
☐ Synchronize both

Figure 68. Graphic user interface of moowinx.
The tool includes a pattern browser (1), which shows an animated preview (2) and a description (3) of all available patterns. The user is taken step by step to define the specific configuration of the pattern. Graphic elements involved in a pattern are visually linked by behaviour icons (4) and the behaviour settings can later be edited (5).

4.2.2 Basic architecture

In its original design moowinx was a timeline-based authoring tool. The content of a course is placed on slides of a timeline. The timeline is divided into discrete positions. These positions are called pages, slides or frames (frames of a movie). A virtual playhead can be moved to one of the discrete positions. All objects that are located at the current position of the playhead are displayed on screen.

4.2.2.1 Content in a multi-layered timeline

To organize the content along the timeline, a timeline can consist of several layers and each layer can consist of several slides. A slide defines a section that spreads over one or more timeline positions. The content of a slide that spreads along the whole timeline is visible all the time. The content of a slide that spreads over one position is only visible if the playhead is placed on that position. Thus, how long content objects are visible depends on the length of the slide they are placed in. A background layer may contain only a single slide that is visible all the time (as a "master slide"). A content layer may have different slides at each position. This allows flipping through content pages. Additional layers can be used to divide the space along the timeline into chapters, replacing content objects chapter by chapter rather than page by page.

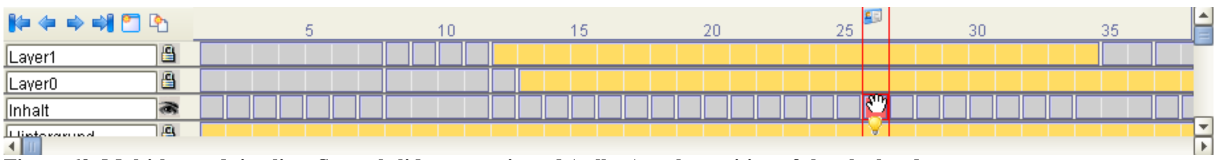


Figure 69. Multi-layered timeline. Several slides are activated (yellow) at the position of the playhead.

4.2.2.2 Content objects

A content object is considered as a basic visual element such as a text label, a box with formatted text, a shape, an image, an input field etc. The visual appearance of an object is defined by several properties such as location, size, colour, text string, opacity etc.

Dynamic behaviour is based on replacing content objects (depending on the position of the playhead) and altering the properties of content objects (e.g., animation of size or location of an object). Interactivity is based on influencing the playhead position and property changes by user inputs such as keystrokes, mouse motion or dragging of objects.

4.2.2.3 Interactive properties

In its early version moowinx offered several ways to authors to define the dynamics and interactivities of the content elements when the file is used.

Dynamic changes included:

- Stepping through content by placing the playhead on the next, previous, or specific page
- Cel animation by automatically run the playhead through slides with slightly different content
- Key frame animation by interpolating the positions of an object that is placed on two succeeding slides
- Scripts to change property values of content objects such as location, size, or colour

Interaction options included:

- Controlling the playhead position via buttons (step forward, step backward, run, stop)
- Direct manipulation: grabbing an object and dragging it to another position, entering text into an input field
- Definition of scripts that are executed on user input such as mouse events

Each object could have several properties that define reactions to user input as a script. For example, the properties mPressed, mEntered and mReleased define scripts that are executed if the user presses the mouse button on an object (mPressed), the mouse button enters the object (mEntered), or the mouse button is released on that object (mReleased).

The scripts are simple text strings that define a program in Tcl. Tcl is a scripting language that can be extended with new commands. Thus, we created special commands to change the properties of an object on a slide by addressing it via a unique name. Properties could be changed directly (e.g. setting the x and y coordinates of an object) or using an animation command that changed the property in several steps over time.

4.2.3 Implementation of the basic architecture

moowinx is implemented in Java (except for some export libraries to generate Flash files). Each of the elements of a course has a corresponding class definition: TAPTimeline, Layer, Slide, TAPElement, and TAPProperty. There is one singleton instance of TAPTimeline that holds a list of Layer objects; each Layer object holds a list of Slide objects; each Slide object holds a list of TAPElement objects.

TAPElement is an interface that can be implemented to create new element types (types of content objects). Third party developers can implement the TAPElement interface. New element types can be placed in a plug-in folder that is checked on start-up. Thus, new element types can be added without recompilation of the whole class hierarchy. An implementation of TAPElement defines the model of a content object. The changeable properties need to be defined by implementations of the TAPProperty interface. Implementations of TAPProperty provide value ranges and strategies to manipulate the property in an object inspector (e.g. colour properties are set by opening a colour dialog). Usually there are several properties that belong together such as the x and y for coordinates, width and height for size, red, green, blue for colour, and mPressed, mClicked etc. to define action scripts. Therefore, properties are grouped by implementations of TAPPropertyGroup. A list of available TAPPropertyGroups for a specific element type can be achieved through the TAPElement interface.

The model of a TAPElement is represented by its properties. The view of a TAPElement is represented by a JComponent implementation. JComponent is the base class for visual components in Java’s graphic user interface framework Swing. A TAPElement can use an existing component such as JButton or JLabel or use a new sub-class of JComponent.

A TAPElement implementation has no direct controllers (though the JComponent implementation may have). The only legal way to change a property of an TAPElement is a call of TAPElement.setProperty(). For example, mouse listeners added to the JComponent could set new location properties if the user drags the component on the screen. There are also specific Tcl commands in moowinx that change the properties of a TAPElement via its setProperty() method.

4.2.4 Motivation for a visual language

The first version of moowinx only allowed authors to define interactive behaviour by Tcl scripts. A Tcl script was executed if an event occurred in one of the content objects. There were also Tcl scripts for slide events such as onEnter (invoked when the playhead entered a slide) or onRepeat (invoked in time periods when the playhead stayed in a slide).

4.2.4.1 Challenges of text-based programming

While small scripts could account for many interactions from simple replacements of images to physical simulations, most end-users were not programmers. Typing scripts requires the user to (Kohls, 2004):

- Know the available commands
- Know the exact syntax of a command
- Know about the order and semantic meaning of each argument
- Abstract from the visual meaning of argument values (i.e. entering the colour values textually instead of using a colour chooser dialog)
- Remember element and property names to access them
- Strictly avoid any type errors

4.2.4.2 Promises of visual programmig

These problems could be addressed by a visual programming language (Schiffer, 1998). An early attempt was an implementation that simply represented a Tcl script visually. There were several advantages to this approach:

- Instead of typing the command names, commands were selected from menus. Thus users could see which commands are available and syntax errors were avoided.
- The visual representation offers a dialog to a edit all parameters. The order of parameter is irrelevant, the users see which parameters are needed and the values can be checked instantly.
- Commands that have a visual impact (such as move, resize, show, hide or change colour) could be represented visually by showing the new colour or the new location of an object.
- A Snapshot function allowed capturing the current values of an object as parameters of a command.

The real problem, however, remained unresolved: the hard part of programming is not the syntax but the semantic of a program. A visual representation simplified the syntax and showed the semantic meaning of single commands (e.g. the new colour to be set was shown). But the visual representation did not simplify the semantic of the control structures and program flows behind complex interactive behaviour.

4.2.5 Patterns of program code

As we were creating our own presentations with moowinx we realized that we used some interactions more frequently. Actually, we ended up programming the same interactions in Tcl again and again with only little variations (i.e. different images or different locations were used in the setProperty commands).

4.2.5.1 Synchronize object motion

A nice effect to uncover information was to draw a curtain and show what was behind. To synchronize the motion of the left and right side of the curtain, a few program statements had to be coded for each application.

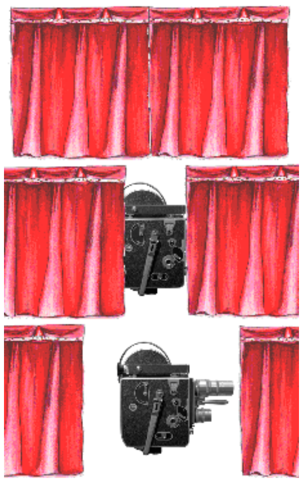


Figure 70. Hiding objects.

Technically we had used two image elements, each representing one half of the curtain. If one side of the curtain was dragged the mDragged actions script was executed:

```
# Get the current position of the dragged element
set xpos [getProperty $thisSlide $thisElement x]

#Compute the difference to its original position
set diff [expr $xpos-49.7 ]

#Apply the same offset to the other element
setProperty $thisSlide leftCurtain x [expr 16.6 - $diff]

#Reset the y location of the dragged element to hold its vertical position
setProperty $thisSlide $thisElement y 28.68
```

Not much code for a programmer but way too much for non-programmers because a lot of abstract programming concepts are involved. But even for programmers this script is very inconvenient because the original positions of the elements are hard-coded into the script. If the author moves the curtain images to other locations, he needs to update the script.

4.2.5.2 Rules and constraints to define a program

If we take a closer look at what really happens on the slide we can see two rules at work:
- One image element should be dragable but only horizontally
- The second image element should move into the opposite direction according to the movements of the first image element

It is much easier to define these simple rules in natural language than programming language. Thus, we had the idea to offer a set of predefined behaviours that could be applied to elements without programming. In their research on end-user programming Pane, Ratanamahatana & Myers (2001) asked students how they would define the game “Pac Man”. None of them specified the rules in a way a programmer would (i.e. properties for locations of Pac Man and ghosts, routines and algorithms). Rather, they used natural language to specify production rules or event-based responses such as “When PacMan eats all the dots, he goes to the next level” (Pane, Ratanamahatana, & Myers, 2001). Many participants also defined constraints such as “PacMan cannot go through a wall.”

For the curtain example, we would need the following behaviours linked to the elements: To the first image element we add a constraining behaviour: “Move only horizontally”. Then we add a motion behaviour to the second image element that links it to the first: “Follow the linked element in opposite direction”.

4.2.5.3 Generalizing the rules and constraints

Because we had seen the effect of two objects moving into opposite directions in other graphics as well (e.g. a channel that opens) a generic description was searched. The core of this specific interaction design was that the motion of one object caused another object to move into the opposite direction. With the example of the lifting block in mind, it showed that "opposite direction" was only a setting in the specific design. Other rotations in motion are useful configurations as well, for instance object A could move leftwards and cause object B to move upwards.

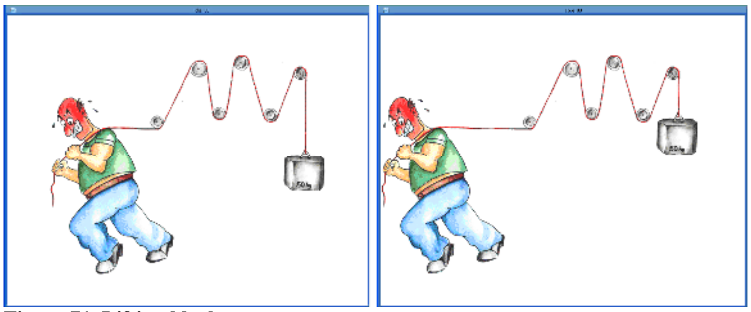


Figure 71. Lifting block.

It was import to mine other variations, that is to find the invariant slots, variables and their default configuration and value constraints. For example, there are graphics which share the same motion relation between objects but the motion translation differs in motion speed and time of start. In fact, all scroll panels are examples of this motion synchronisation: a small object navigates a larger object in a clipping frame. Using such behaviour relations in our tool, we had a chance to define interactive graphics without any programming.

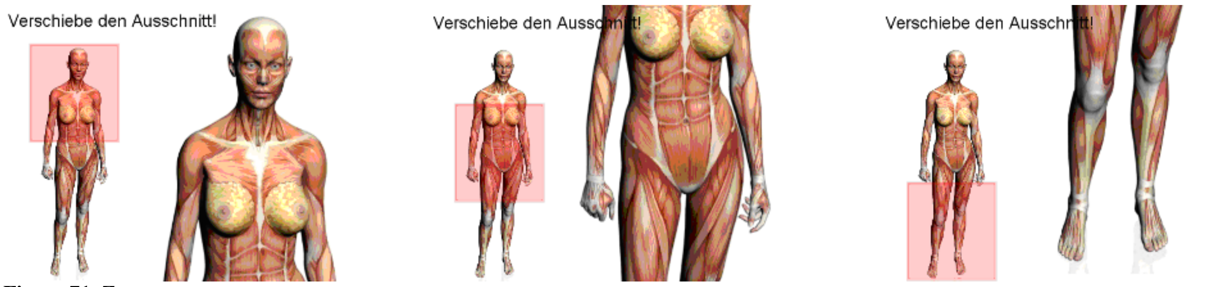

Verschiebe den Ausschnitt!
Verschiebe den Ausschnitt!
Verschiebe den Ausschnitt!

Figure 71. Zoom.

Behaviours are a concept that combines events, predefined calculations and actions in one single rule that can be represented visually. Instead of writing Tcl code as an event script, a behaviour implicitly defines the events and actions that matter for it. It also specifies a reaction to these events. The reaction can be simple or involve calculations and use property values of other elements. To adjust the reaction, behaviours can be configured.

4.2.6 Pattern mining

4.2.6.1 Patterns from experience

We scanned our own works and considered all interactions in the hope of finding more of such recurring interactions that could be expressed in a generic way yet understandable in natural language. About 20 such behaviours were found. Some were already commonly known, such as roll-over or 'drag and drop', others were present mostly in our own designs, for example ACTIVE AREAS that set properties of dragged objects according to background objects. Figure 73 shows an example illustrating a flag (the dragged object), which changes its visual appearance when it is dragged over another country (the active area). The full pattern descriptions can be found in appendix E.

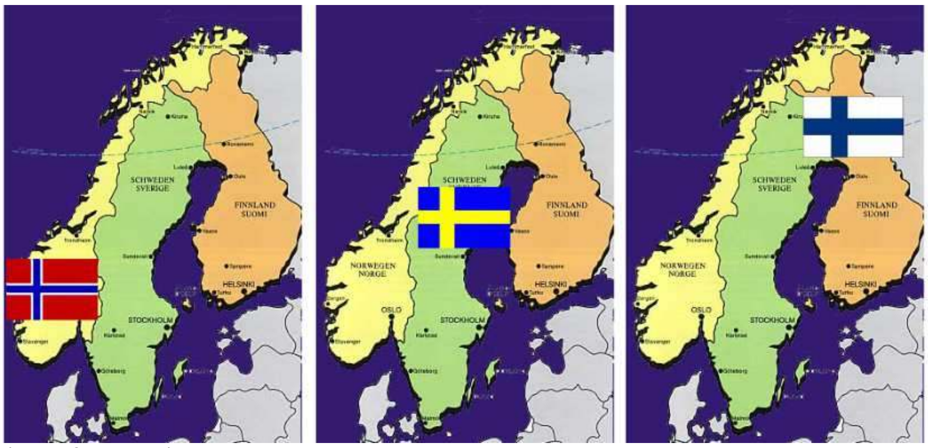


Figure 73. Interactive map.

4.2.6.2 Patterns, but not design patterns

It is notable that the behaviours were not yet complete design patterns in the sense we discussed previously. They only cover the solution part. Having such ready-to-use solutions in an authoring tool helps avoid programming. However, what has not been captured at that point were the core problems tackled by those solutions and the contexts in which they could be advantageously applied. That is, the behaviours are just recurring designs – whether they are good or bad designs depends on the information and the message one intends to deliver by an interactive graphic – the context. For instance, the same interaction principle used for flags that labelled countries, is misused in the example shown in figure 74.

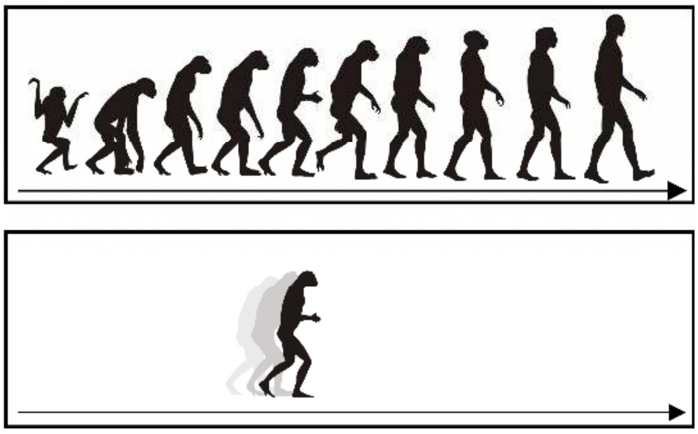


Figure 74. Illustration of evolution.

To illustrate evolution, a “classic” static graphic (upper box) shows a man in progressing evolutionary states on a timeline. In an interactive graphic (lower box) the man is a dragable object. Depending on its position at the timeline, the man is shown in a different evolutionary state. Hence, the man changes dynamically when dragged along the timeline. In practice, this gives a nice visual effect and it could certainly be used as a motivating opener in a presentation. But typical benefits of small multiple representations - such as seeing the progress immediately, getting the chance to compare each state and having visual access to each representation (Tufte, 1990) - are lost. The educational value actually decreases unless the only intention was to motivate.

4.2.6.3 The need for design patterns

It was this realization that pointed out that complete design patterns were needed, which tell both sides of the story: the solution and the context in which it solves a problem (more on the description of the patterns in section 4.3). Actually, the behaviours concept made it so simple to create interactive graphics that we felt a responsibility to attach information about the why and when to each solution. In pattern culture, there is a tradition of reasoning about applicability and consequences in design decisions that was needed for a thoughtful use of interactive graphics in e-learning materials.

4.2.6.4 Patterns from analysis

We had identified complete design patterns consisting of name, context, forces, problem, examples and the core of each solution. But did our patterns matter outside of our own project? That is, did other producers of multimedia content use the same patterns in their applications? Or did they use even more patterns, recurring designs that we had not experienced ourselves and therefore did not mine as patterns? To answer these questions we did a systematic review of more than 700 interactive graphics. The graphics were taken from e-learning titles that were either very popular (best selling titles), best practice (award winners, showcases), or had strong marketing (brands, interactive features were advertised). To consider a wide range of different visualization types, the titles differed in domain (e.g. science, language, art), target group (e.g. K-12, higher education, training), publishers (e.g. textbook publishers, mass media) and medium (e.g. CD-ROM, web based training, interactive whiteboard content).

4.2.6.5 Recurrent relations of input, model changes, and output

To analyze an interactive screen and find recurring structures the interaction was transformed into a structural representation. This transformation was done by considering both visual objects and input/output operations as entities that were related to each other.

Input entities included:
Primitive input: mouse clicks (single, double, right mouse button), mouse motion (with/without pressed button), mouse pointer enters or exits an area, mouse drags an object, direct property manipulation (e.g. resize) etc.
Spatial input: relative positions of objects to each other, arrangement of objects, relative size of objects, object collision, drag and drop, object A is contained in object B, etc.
Composed input: A sequence of primitive and/or spatial inputs, e.g. click sequences or collisions sequences of multiple objects

Output entities included:
Change of visual properties: set image sources, set visibility of objects, change of size, position, colour, opacity, rotation etc.
Change of system mode: velocity of objects, change of input mode (e.g. dragging a mouse pointer can either group objects or create new objects depending on the mode), different object reactions for further inputs etc.
Object creation: Creation, destruction or duplication of objects, definition of object relations etc.

4.2.6.6 Abstraction of recurrent interactive graphics

In transforming an interactive graphic into a more abstract representation, a method was found to describe the interaction rules and roles of objects independent of their superficial features. That is, on this level it did not matter whether a pair of curtains or the doors of a channel moved as long as the objects behaved in the same way.

The strategy to analyze the selected e-learning titles was as follows:
- For each title the content was scanned for interactive screens.
- For each interactive screen the interaction rules and roles of visual elements were analyzed.
- If both rules and roles matched to a proposed pattern candidate, the candidate was strengthened.
- If no matching candidate was found, a new candidate was proposed.
- If a candidate had been strengthened several times, it became established.

The result of the search was a list of established candidates. Established candidates described interaction techniques that can be found in several e-learning products available on the market (respecting the “Rule of Three”). However, frequently using the same solution is not a proof of quality. Therefore, an established candidate only becomes a member of the pattern collection if, furthermore, there is evidence that the interaction type supports principles of cognitive science, learning psychology, instructional design, or usability. This evidence is made explicit in the rationale field of our description format (see section 4.4.1.2 about the different formats used).

It turned out that most of the patterns we had mined in our own work were also present in other multimedia content. In addition, we discovered some new patterns that we added to our collection. We also widened our understanding of the contexts in which a certain pattern could work.

4.2.7 Implementation of patterns

The different patterns are implemented as sub-classes of the Behaviour class. Conceptually, a Behaviour is a response to user input. It consists of input data (an event), a state (model), rules to process the data, and output data (changes to slides or content objects). Each Behaviour represents a pattern of interacting content objects on a slide of moowinx. Hence, a Behaviour has a list of TAPElements that participate in the pattern. Each element can have a different role.

4.2.7.1 Adding behaviours at edit time

At edit time, an author can add new Behaviours and change the model of each Behaviour. A Behaviour always has an iconic graphical representation on stage. The icon is linked to the elements on stage that participate in the behaviour. To change the model, a user can add further elements that participate, remove elements, change the roles of element, or set-up configuration properties for the Behaviour.

Example: The pattern Synchronize Objects has a guiding object whose motions are imitated by fellow objects (see description and diagram in appendix E). The implementation of this pattern in moowinx is located in SyncObjectBehaviour, a sub class of Behaviour. Once the behaviour is added to some elements on a slide an icon connects the participating elements. The bold line connects the guiding object to the icon. The other lines connect the Fellow objects to the icon. A user can add or remove Fellows through the visual interface. He can also drag the bold line to another element on stage in order to change the Guide object. Adding, removing or changing the list of participants changes the model of the behaviour. By double-clicking on the icon, a dialog opens that allows further set-up. The way the Fellow objects imitate the motion of the Guide can be changed: the motion can be delayed, rotated (e.g., move to the opposite), or accelerated.

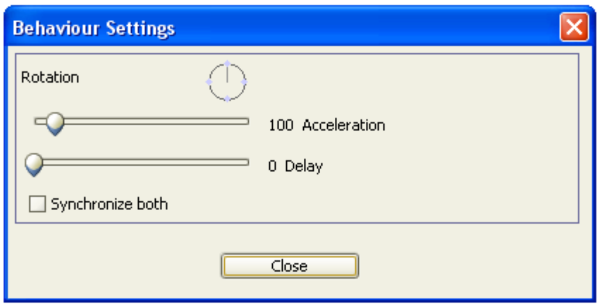


Figure 75. Changing the settings of a Behaviour

4.2.7.2 Performance of behaviours at runtime

At runtime, a Behaviour listens to events that are important for the participating objects. Typical events are mouse events, property changes, or the motion of participating objects. On starting a presentation, all Behaviour objects are notified. A specific Behaviour, then, can initialize the event listeners for all TAPElements that are part of its model. Very often this simply means to add Java Swing's mouse listeners to the Swing representation of a TAPElement (i.e., a JComponent that is the view of a TAPElement).

4.2.7.3 Geometric events

A special class of events are geometric events such as intersection or inclusion of objects. Such events are not part of the standard graphic user interface framework of Java. Therefore, we implemented a GeoEngine for moowinx that observes the locations and bounds of all visual elements on a slide. The engine checks for each object that has changed its location, size or shape whether it spatially intersects other elements, includes other elements, or is included by other elements. If the relation to another object changes (e.g. an object starts or stops to intersect an other element), a GeoEvent will be fired.

For all geometrical events a TAP.visuallang.GeoEvent object will be created and sent to the registered listeners (Behaviour objects that implement the listener interface). The GeoEvent stores:

- the motion vector that led to the spatial change which triggered the event
- a reference to the element that triggered the event (that is the element which has moved)
- a reference to the involved element (the element that has not moved but is affected)
- a reference to the element that actually distributes the event (owner element)
- for containment events there are also references to the element container and the contained element

If two elements start or stop to intersect for both elements an intersection event will be fired and distributed to all registered GeoIntersectionListener objects of the triggering TAPElement and to all GeoIntersectionListener of the involved TAPElement. The GeoIntersectionListener listens to start and stop intersection events:

```
public interface GeoIntersectionListener {
    public void startIntersection (GeoEvent e);
    public void stopIntersection (GeoEvent e);
}
```

The GeoContainListener interface covers events of inclusion:

```
public interface GeoContainListener {
    public void startContains (GeoEvent e);
    public void stopContains (GeoEvent e);

    public void startContained (GeoEvent e);
    public void stopContained (GeoEvent e);

}
```

Example:
The pattern DROP ACCEPTOR ensures that no more than a maximum number of objects are dragged over another object. On starting the presentation, the behaviour registers itself as a GeoContainListener to the GeoEngine. If one of the participating TAPElements is dragged to the container element, the behaviour checks whether the maximum number is exceeded. If that is the case, the element will be moved out of the container automatically.

4.2.7.4 Output of behaviours

Based on its current model state and the events it receives, a behaviour generates an output by changing the properties of its participating elements and its own state. For example, if the SyncObjectsBehaviour receives an event that the Guide object has moved 10 units to the left, and 5 units to the top, it applies the same motion vector (or a modification) to all participating Fellow objects.

An example for the change of a behaviour model is the Drop Acceptor which changes its own state by increasing a counter when another object is dragged into the container object.

Not all outputs trigger or block motions of other elements. A switch behaviour switches elements between ON and OFF states by storing alternative property values. Thus, if the switch behaviour receives an event that switches an element to ON, then its output is to set the properties of the element to the values of the ON state (e.g. set the image or colour) and the model of the behaviour has to save that the switch is in ON state.

Motion outputs, however, need a special treatment because a new location for an object should not be set directly. For example, the SyncObjectBehaviour may try to move an object 10 units to the left. Yet there could be another element that is part of a NoGoArea behaviour and blocks other elements to move to that location. Moreover, the SyncObjectBehaviour could involve an object that also participates in a PushBehaviour; such an object pushes other object on the stage. However, if one of the pushed objects is blocked by a No-Go-Area, the pushing object will be blocked indirectly. That means that behaviours can influence each other and new interactions of elements emerge. To account for the constraints and side effects of other behaviours, a motion is never applied directly by changes of x and y coordinates. Rather, the location change is represented by a motion vector that is send to a BehaviourEngine. This behaviour engine treats the motion vectors like "as if" motion – it tests what would happen if the motion would be applied and gives other behaviours the chance to modify the motion vector.

4.2.8 The behaviour engine

Each change of an object's position is triggered by motion vectors. This includes dragging with the mouse pointer. Rather than directly setting a new position if an object is dragged, a vector is generated that points to the new target position.

The behaviour engine is started for an object that is about to move. The move is represented by a motion request; this request is a vector requestedVector (dx, dy). The vector is sent to the object that is requested to move. The behaviours that are connected to that element can transform the requestedVector according to their constraints. There are two kinds of transformations: conditioned and unconditioned. Unconditioned transformations always change the requestedVector. Conditioned transformations only apply if the motion according to the requestedVector would cause a collision with other objects; the vector is split at the collision point into two parts and for each part it is tested whether the motion is possible.

4.2.8.1 Creation of a motion request

A requestedVector is created if:

- the user drags an object with the mouse pointer
- an animation step is applied
- there are side effects of the motion of other objects, e.g.
    - The collision of a bulldozer generates a motion request for the collided object
    - Motion listeners (e.g. transportation, motion synchronization) observe motions of other objects

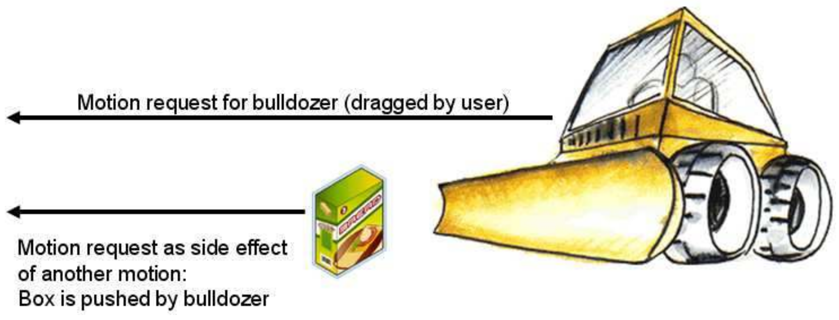

Motion request for bulldozer (dragged by user)
Motion request as side effect of another motion: Box is pushed by bulldozer

Figure 76. Creation of motion request.

4.2.8.2 The behaviour engine starts its works

A motion request is sent as a requestedVector to an object A. First, the engine checks whether A is blocked. If A is blocked, the request will be ignored. If A is not blocked, the request is accepted and A will be blocked (to avoid multiple requests and recursions, e.g. A triggers motion of B and B triggers motion of A).

4.2.8.3 Unconditioned transformation

Unconditioned transformations are always performed. They change the original motion of an object. For example, the DragRestriction-Behaviour changes the motion in a way that the object always moves along a straight line. For the horizontal drag restriction, the y-component of the requestedVector is set to 0 because only horizontal motions are allowed.

4.2.8.4 Transforming the motion request

The engine tries to satisfy the requestedVector as far as possible, i.e. to apply the whole vector. However, some behaviours constrain the motion of objects (e.g. No-Go-Areas), which restricts the motion that can actually be applied. Motions that can be applied without constraints or side effects are stored in an applyVector. Motions that may be changed or cause other elements to change are stored in a remainingVector.

At the beginning the remainVector is the same as the requestedVector. A main loop runs through all behaviours that might change the remainingVector. The loop works on the remainingVector until

- remainingVector == (0,0): no segment is left
- remainingVector$_t$ == remainingVector$_{t-1}$: the remaingVector cannot be shortened, i.e. the behaviour constraints allow no more motion

The goal of the main loop is to calculate which parts of the requestedVector can be applied. Several checks are done in this loop.

4.2.8.5 Motion without collision

The first check is whether motion according to the requestedVector can be applied without any collision. If that is possible, the complete motion can be applied in one step. Hence, the applyVector is equal to the remainingVector. The complete remaining vector segment is executed. The remainingVector is set to (0,0) because no further segment is left.

4.2.8.6 Motion with collision

If a motion would cause a collision (note that the engine runs in "what if" mode without changing any properties for real) the motion needs to be treated differently.

Motion constraints only apply from the point of collision. For example, a motion request can be performed up to the point where the object collides with a No-Go-Area. That means object A can move without problems up to the point of collision. Therefore, the remainingVector is split into two parts:

- The applyVector defines the segment that object A can move without collision
- The remainingVector stores the remaining segment:
  $\text{remainingVector}_t = \text{remainingVector}_{t-1} - \text{applyVectory}$

The applyVector can be executed directly similar to the case without collision

4.2.8.7 Behaviour constraints

The remainingVector now contains a segment of motion that is subject to the constraints of the behaviour(s) interested in the collision. A behaviour can transform the remaining vector according to its rules. Some examples may illustrate the principle:

- A No-Go-Behaviour of object B defines that object A is not allowed to be dragged over B. Hence, the remaining motion request is not possible to execute. The No-Go-Behaviour omits the remaining motion by setting the remainingVector to (0, 0).
- The Bulldozer behaviour pushes a collided object B as far as object A can continue its own motion. The behaviour generates a new motion request for B. The requestedVector for B is the remainingVector of A. Object B should be pushed away to open the space for object A. The motion request for B starts another run of the behaviour engine, this time for object B. The engine returns as a result the actually performed motion vector (see below). If B happens to collide to a No-Go-Area of object C, then B cannot move as far as requested. Hence, element A cannot move the full segment that remained for A: object C first stops B and then B stops A.
  What does that mean for the remainingVector of A? Object A can only move as far as other objects allow. Hence, the remainingVector is set to the actually applied vector of B.

How is this handled by the engine? The remainingVector is the favoured motion for object A; object A should move along this vector if possible. Each behaviour that defines constraints for colliding objects can change the remainingVector. Each behaviour gets the chance to transform the remainingVector. The element iterates over all its linked behaviours and lets each behaviour transform the remainingVector if needed.

The problem is that behaviours can compete. For example, should a No-Go-Area stop a bulldozer or should a bulldozer push the No-Go-Area? Therefore, each behaviour gets a different priority value that defines which behaviours first get the chance to change the remainingVector.

If the remaining vector is sent to all behaviours but does not change any more, the main loop can terminate and the applyVector can be executed by effectively changing the location properties of object A. Also, object A gets unlocked and is ready to accept new motion requests.

4.2.9 Towards wizards

The behaviours allow end users to create interactions of elements without programming. In trainings we observed that users picked up the concept very quickly and produced interactive graphics after 15 minutes. However, the behaviours were still not self explanatory and needed guided instructions. A user needs to know about the roles involved in the behaviour, i.e. what elements are needed, which effects the elements have, and which configuration options there are. For example, for the SyncObjectsBehaviour, a user had to first add at least two objects to the stage, and select both objects and choose the SyncObjectsBehaviour from the menu. Moreover, he had to understand that one object can be moved freely and the other objects follow the motions.

4.2.9.1 Implicit knowledge in wizards

There are also more complex configurations and behaviours. For example, an Active Area changes the appearance of an object that moves over different areas. Each area is an object in the background. In order to set-up the behaviour, a user first has to create all background objects before the behaviour can be added. However, without knowing the behaviour and its functionality one would not create such objects in the background. It is like the chicken and egg problem: without using the behaviour one will not know which objects are required; without the objects required one cannot use the behaviour.

The written documentations of patterns are one way to inform users about the possibilities of behaviours and which objects are required. To solve this problem directly in the authoring tool, we specified wizards that generate the behaviour by creating all required objects, choosing meaningful configurations and adding the behaviour to the stage finally.

4.2.9.2 Wizard browser

To use a wizard, a user first opens a wizard browser. This browser contains an animated example of the behaviour and a brief description. The benefit of the browser is that a user can pick the appropriate behaviour and learn about all behaviours available and their contexts of application. The behaviours are clustered into the categories “Moving elements”, “Restrictions”, “Highlights”, “Switches”, “Changing images”, “Business”, “Navigations”, “Animations”, “Moving indicators”, “Activators”, and “Drag & Drop”.

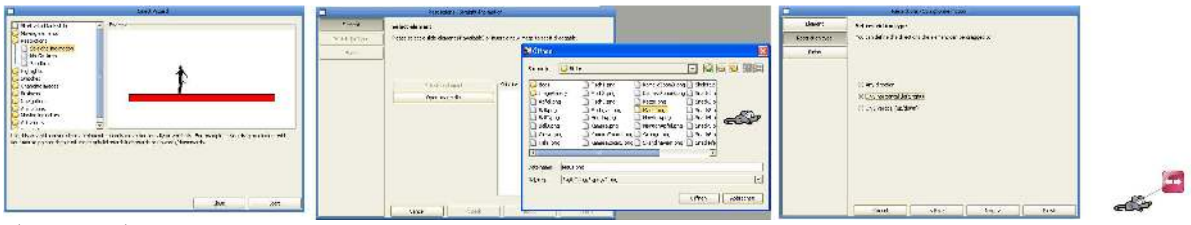


Figure 77. Wizard browser

Once a wizard is chosen it guides the user step by step through the process of setting up the behaviour. Each wizard roughly follows this sequence:

- Pick-up the objects that participate in the behaviour
- Set up configurations for the objects if needed
- Set up general configuration for the behaviour

This sequence can unfold into several steps. If there are different roles of participating elements, each role is assigned in a different step. If there are multiple configuration variables to be defined, each variable will be set in a different step. The benefit is that on each step the role or variable can be explained to the user.

To pick up an object, the user can always select an existing object on the screen (e.g. text, images or shapes) or create a new object. Thus, a user does not need to load images in advance without knowing their roles. A wizard offers to create the object types most appropriate for that role, i.e. to open an image for buttons, enter text for labels, or draw polygons to define areas.

Many behaviours allow multiple objects to have the same role (e.g. a SyncBehaviour can have multiple fellow objects). In this case, a wizard allows selecting multiple objects from the stage, or creating multiple new objects.

Sometimes a behaviour stores configurations for each participating object. Switch behaviours store ON and OFF states for each object that can be switched. Hence, a wizard allows to set the alternative states for each object directly on creating the behaviour.

The last step of the wizard is always to create the behaviour. In this step, all objects that have been specified in the previous steps (such as images or text labels) are created. Then, the behaviour is added to the slide and linked to the objects, assigning the roles accordingly.

4.2.9.3 Wizards and patterns

There are more wizards available than behaviours. A wizard creates a behaviour, the participating objects, and sets a specific configuration. This means different wizards can use the same behaviour but in different ways, i.e. offer different object types in the creation steps or preset standard configurations. For example, the ACTIVE AREA pattern has an object that changes depending on areas in the background. There are two wizards that both use the same behaviour; however, one wizard offers the creation of a text label and the other wizard creates an image that changes over the areas. Likewise, for switch buttons there are different wizards that use the same behaviour but set different triggers (mouse over vs. mouse clicks) to switch the state.

Specific behaviour configurations that recur frequently mean that we have a specialized pattern. Switching objects ON and OFF is a pattern; switching an object ON and OFF by clicking on the object is a specialized pattern; switching an object ON and OFF by mouse roll-overs is yet another specialization. Technically the two pattern specializations are represented by the same behaviour class because they differ only in one variable: whether the switch is triggered by mouse clicks or mouse rollovers. The behaviour adds different mouse listeners to the object accordingly but the processing and output of the behaviour are always the same.

For end users, however, it makes a big difference whether a switch is triggered by clicks or roll-overs:

- roll-overs coincidently change objects whereas clicks require deliberate actions,
- roll-overs can switch objects to ON only temporarily and the mouse pointer cannot be used somewhere else on the screen, and
- touch devices such as interactive whiteboards or tablets do not support roll-overs directly (see appendix E for more differences between the button types)

The two specialized patterns are used for different purposes. They are, as our experiments have shown (see section 4.3), perceived as different categories of interaction. They require different elaborations when documented as design patterns because they resolve different problems and have different consequences (see section 4.4). Only the implementation of the patterns is (almost) the same.

Wizards are a way to account for specialized patterns without specialized behaviour implementations. Since wizards guide through the configuration step by step they are a means to differentiate a more abstract pattern in a specific way. Each of the wizard steps unfolds the general pattern into a particular behaviour with a set-up configuration.

Wizards can also work the other way around. Instead of particularizing a pattern (or behaviour), it can build up more complex patterns based on existing patterns (or behaviours). For example, there is a wizard that generates a curtain that can be dragged open. It is the curtain that originally inspired the behaviour idea. We had seen that the interaction of two curtain halves requires two behaviours: a horizontal drag restriction and a synchronization of motion. The curtain wizard creates two image objects and both behaviours. The interplay of the behaviours let a new complex behaviour emerge.

Another example is “Multi-element race” which creates multiple objects and connects any two objects with a sync behaviour that imitates motion with a delay. If the first object is moved, the second object follows shortly after; the third object follows the second shortly after, and so on. The number of behaviours created depends on the number of objects picked up in the wizard. For each motion relation one behaviour is instantiated.

4.2.10 Lessons learnt

4.2.10.1 Emergence

It is important to understand that the wizards and behaviours are not just templates. A template is a specific configuration of patterns that can be instanced several times (see section 3.2.11.5 on images and templates). Wizards and behaviours, however, are still open for the combination with other patterns. A wizard allows many variations by freely picking up objects and setting the configuration. Behaviours are even more open because each object can be part of multiple behaviours. By the combination of several interaction patterns new patterns emerge because the behaviours influence each other. One behaviour "acts" differently if other behaviours are around.

The downside of these emergent qualities is that end users can set-up configurations that do not function properly. For example, by chaining several objects by motion synchronization, we can easily set-up an infinite loop. A programmer may understand the effects but a non-programmer should not bother about infinite loops. moowinx traps typical situations that lead to infinite loops but the openness of configurations may create situations that are not safe. Not all emergent behaviour is favourable.

4.2.10.2 Patterns in different interaction spaces

The combination of patterns is usually without danger if the patterns act in "different spaces", i.e. their interactions are subject to different dimensions. For example, patterns that change the colour or image of an object do not interfere with patterns that change the geometrical location of an object. Motion synchronization of objects happens in a different space than organizing the ON and OFF states of objects – even if the same objects are involved. However, if several motion and motion restriction patterns involve the same elements, erroneous configurations are possible. Likewise if multiple switch behaviours act on the same objects there could be contradictions (e.g., an object that switches ON multiple objects that are also part of radio button group). Therefore such attempts are blocked by moowinx: each object can only participate in one switch behaviour.

4.2.10.3 Universals

While there are many ways to categorize the patterns and find patterns of different levels, each pattern is a universal. This can be shown most obviously for behaviours. A behaviour exactly specifies the relations between the participating elements. Although two instances of a behaviour can look very different (e.g. the use of different images can have very different meanings), the structure is the same. In fact, two instances of a behaviour use the very same code (physically in the computer memory) to organize the interplay of their participants.

Wizards also follow the same universal sequence in generating very "different" interactions. For example, the configuration of the synchronization behaviour has very different effects depending on the parameters: a rotation of 180 degree moves objects into opposite directions. A rotation of 90 degrees can show the motion of a lifting block. Wizards are recurrent ways of modifying patterns or combining patterns to more complex patterns. Thus, wizards cannot exist independently but in combination with other patterns they create new patterns.

4.2.10.4 Constrained generating procedures

Both wizards and behaviours are examples for constrained generating procedures (see section 3.2.11.3). A wizard is a building block to generate a proper configuration of a behaviour and its required participating objects. A behaviour is a building block that generates the interactive interplay of objects at runtime. The input of a behaviour (events) along with its state define its output; this output (e.g. motion) can be the input for other behaviours. A formal definition of each behaviour can be stated as finite deterministic automata (Kohls, 2004). A finite deterministic automaton is a special case of a constrained generating procedure (Holland, 1998).

A behaviour automaton is a very small finite deterministic automaton specified by the tuple $A = (I , O, Q , d , q_0, F)$. Where I is the input alphabet, O the output alphabet, Q is a set of start states, d a set of a transition function that maps input symbols and current states to a next state, q_0 is the start state and F is a set of terminal states.

The input alphabet of a behaviour automaton can contain any of the event types supported by moowinx as symbols. For example a behaviour that responds to startIntersection and to propertyChanged events has an input alphabet I = {startIntersection , propertyChanged}.

The processing is defined by a transition of the automaton from one state to another. The output of a behaviour is defined by the output alphabet O of a behaviour automaton. The output cannot only be a manipulation of the presentation (such as changes of element properties, start animations), but may also affect the behaviour model the automaton belongs to.

Each configuration of states (the model of a behaviour) and inputs (the events) can be mapped to a new state (the output or effect generated by the behaviour).

4.2.10.5 The problem of classification

The analysis of our own content and of multimedia titles had shown that there are indeed patterns of interactions that recur again and again. The problem, however, was that stable patterns existed at various levels of abstraction and granularity. More general patterns covered more instances but the instances had less in common; more specific patterns were more meaningful but led to an explosion of patterns.

For example, there are different types of switches. A very particular example is a switch of images if the user clicks at the image. On a more abstract level, the switch of images, different colours, opacities or text is the same: a switch between two properties. Another abstraction involves the trigger type: mouse clicks to switch an object ON or OFF, mouse entering to switch an object ON, mouse exiting to switch it OFF, and mouse down to switch it ON, mouse up to switch it OFF. Moreover, the trigger does not need to be the object itself: a click or roll-over of another object could trigger the switch. Another variation included switches that include multiple objects that can be switched ON and OFF: radio buttons, multiple buttons switched ON and OFF together or in sequence.

The patterns ACTIVE AREA and ACTIVATOR are two specializations of a more general pattern. In both cases the intersection of two objects changes the appearance of one object. ACTIVE AREA has multiple objects that change one object that moves over the areas; ACTIVATOR has one object that changes the appearance of different areas when it moves over them. The general pattern is: one object changes another on intersection. On a closer look, this interaction is a special type of a switch. Rather than using mouse clicks or roll-overs as triggers, the intersection of objects is used as a trigger. One of the two intersecting objects changes its state from ON to OFF – just as switch behaviours do. The ACTIVATOR works similar to a radio button because there is always only one element activated (switched ON).

But a very abstract switch behaviour makes no sense. Between an activator and a radio button there are too many differences. In fact there are more differences than commonalities. Though instances of both patterns have a common root – the switching of states – they have unfolded into very different directions. If we had a category of “abstract switches”, one would not know how an instance of that “abstract switch” would look like. It could be a radio button or an active area. However, if a designer speaks of a “roll-over button” one has a more concrete idea in mind. “Roll-over button” implicates what interactions there will be; whereas an “abstract switch” does not reveal whether there will be mouse clicks, mouse motions or motion of objects involved. A “roll-over button” has more gestalt than an “abstract switch”. The pattern “roll-over button”, however, does not fully specify the design of a button either. It is still open enough (gestalt-bar or design-able) that it can unfold into different directions, i.e. which properties change (image or colour) and how the properties change (what the image shows or which colour values to use).

Likewise, we can find hierarchies for the synchronization of object motion. For example, one could have specialized patterns for rotations, delays, and acceleration. One could also consider the TRANSPORT OBJECTS pattern as a special case of SYNCHRONIZATION. Here, the motion vector cannot be modified but the synchronization is conditioned, i.e. an object follows its guide only if it is contained in that guiding object. The push behaviour also synchronizes the motion objects but only if the pushing object intersects the pushed objects.

The wizard browser in moowinx organizes the different behaviours and their specializations into more general categories such as motion, motion restriction, highlighting, fade in/out, change image, navigation, animation, interactive info graphic, activators, and drag & drop. Wizards that use the same behaviours can be found in quite different categories. The reason is that behaviours reflect technical similarity whereas wizards are specific enough to reason about the purpose of an interaction. Thus, the specialized behaviours are ordered by purposes (emergent qualities) rather than technical similarity. However, this order is never objective. In fact, the behaviours and wizards have been re-ordered in several versions of moowinx.

4.3 Induction of patterns – a pattern experiment

To find out whether we have split our patterns at the right level, we designed a study in which novices were asked to identify classes of similar interaction based on a sample of 14 interactive graphics. The graphics were created based on three distinguished patterns. Each graphic implemented only one of the patterns. On a very abstract level, one could say that all of the graphics are members of the same pattern class, some sort of ELEMENT ACTIVATION. That is, trigger events e on input elements $i_1,...,i_n$ cause output elements $o_1,\dots,o_n$ to change property values between $p_{default}$ and $p_1,\dots p_n$. For example, clicking (event e) at an object (input element i_1) on the screen causes a text label (output element o_1) to pop up (p changes from $p_{default}$=invisible to p_1=visible). But there is variation in the type of events (mouse input, drag operations), the number of input and output elements involved, the property types and values that change, whether the sets of input and output elements are disjunctive, and how many output objects can alter property values simultaneously. Because there are more variations than similarities, it makes sense to identify patterns on a more specific level. In our case, we originally identified the patterns STATE SWITCHER, ACTIVATOR and ACTIVE AREAS (see appendix E).

A STATE SWITCHER changes the properties of an output element if a mouse event occurs. A mouse click or rollover on i_x causes o_x to change its properties. ACTIVATOR and ACTIVE AREAS both use an input element that can be dragged over areas on the screen to trigger an intersection event. The difference is that in ACTIVATOR the area in the background changes between two property values whereas in ACTIVE AREAS the dragged input element changes itself using one of n property values. In the first case there are multiple output elements; in the second case the input and output element are the same.

Of course, we could further split up the patterns. If we consider the type of the output object (e.g. text, images, shapes) to be discriminative, there would be more sub patterns. But if there is much variation in discriminative dimensions, the number of sub pattern explodes. Indeed, the decision which dimensions are discriminative is arbitrary, and the judgement can be adequate or not. Therefore, we need to evaluate whether there is support for the proposed pattern partition.

4.3.1 Setting

To test whether the participants identified the same or similar patterns, three sets of interactive graphics were created. An overview of the graphics is given in appendix D. Each set contains graphics that implement one of the patterns: STATE SWITCHERS are in P1 = {A_{P1}, B_{P1}, C_{P1}, D_{P1}, E_{P1} }, ACTIVATORS are in P2={F_{P2}, G_{P2}, H_{P2}, I_{P2} }, and ACTIVE AREAS are in P3 = {J_{P3}, K_{P3}, L_{P3}, M_{P3}, N_{P3} }[28]. The union of the pattern sets is G={A_{P1}, B_{P1}, C_{P1}, D_{P1}, E_{P1}, F_{P2}, G_{P2}, H_{P2}, $I_{P,}$ J_{P3}, K_{P3}, L_{P3}, M_{P3}, N_{P3}}. Figure 78 shows the snapshots of the graphics and to which patterns they belong. The figure also shows possible super and sub patterns. The proposed level of abstraction for the patterns is highlighted by a bold outline.

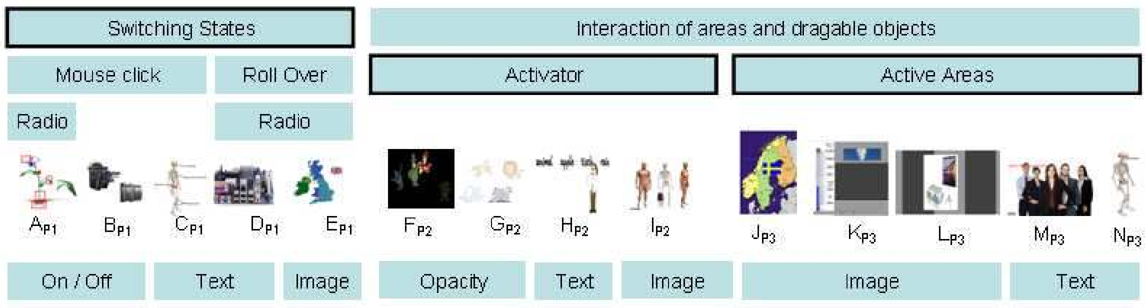


Figure 78. Hierarchy of patterns for interactive graphics.

4.3.1.1 Discriminative features

To discriminate the graphics one can compare the attribute values on multiple dimensions. There are some dimensions related to interactions and these are the ones that make the patterns distinct. They are explained in the following.

[28] This is not the pattern P3 that has been discussed in section 3.2.

Event type: Mouse inputs are basic mouse operations such as clicks, mouse enter, mouse exit or mouse dragging. Drag inputs are operations that change the x,y position of an object. More complex events include the intersection or inclusion of elements.
Activation event: The event that changes a property value of the output element from $p_{default}$ to p_x.
Deactivation event: The event that resets a property value of the output element to $p_{default}$.
Number of dragable elements: How many elements in the graphic can be dragged and change their positions?
Number of input elements: How many elements can receive input events, e.g. how many elements are clickable? The value n indicates that there may be any number 1,..., n of elements that can receive input elements.
Number of output elements: How many elements can change after an event?
Input = Output element: Are input and output element the same element, that is $i_x = o_x$?
Simultaneously activated elements: Can there be multiple elements with property values different to the default values?
Property values per element: How many different property values $p_1,\ldots,p_n$ can a single output element take?

$GraphicIndex_{PatternIndex}$	Graphic name	Event type	Acti-vation event	Deacti-vation event	Number of dragable elements	Number of input elements	Number of output elements	Input = Output element	Simultaneously activated elements	Property values output element	Shape of input element	Type of output elment	Property change	Property values	Function	Domain	Style
A_{P1}	Flower	mouse input	mouse click	mouse click	0	n	n	X	1	2	Rectangles, red outline	image	visibility	Zoomed area on/off	Zoom	Biology, Plants, Nature	Illustration
B_{P1}	Camera	mouse input	mouse click	mouse click	0	n	n	X	n	2	Rectangles, violet, transparently filled	image	visibility	Zoomed area on/off	Zoom	Technology, Historical Object	Photography
C_{P1}	Skeleton (Single labels)	mouse input	mouse click	mouse click	0	n	n	X	n	2	Areas of skeleton parts	text	visibility	labels of bones on/off	Labeling	Medicine, Anatomy, Human (creature)	Illustration
D_{P1}	Computer Board	mouse input	mouse enter	mouse exit	0	n	n	X	1	2	Parts of the board	text bubble	visibility	bubbles on/off	Labeling	Technology, Computer	Photography
E_{P1}	Map (UK)	mouse input	mouse enter	mouse exit	0	n	1	X	1	n	Boundaries in map	image	replace	national flags	Labeling (indirectly)	Geography, Maps, Flags, UK, States	Map (diagram)
F_{P2}	Night watchman	drag input	start interse ction	stop intersect ion	1	1	n	X	1	2	Bodies	image	opacity	20% / 100%	Focussing	Fantasy creatures	Comic
G_{P2}	Animals	drag input	start interse ction	stop intersect ion	1	1	n	X	1	2	Bodies	image, text	opacity, visibility	20% / 100%, on/off	Focussing	Animal creatures	Comic
H_{P2}	Vocabular y	drag input	start interse ction	stop intersect ion	1	1	n	X	1	2	Bounding box of text	text	replace	German word/ English word	Translate text (transforma tion)	Language, Vocabulary	Comic
I_{P2}	X-Rays	drag input	start interse ction	stop intersect ion	1	1	n	X	1	2	Bodies	image	replace	Muscles / Skeleton	View inside (transforma tion)	Medicine, X-Rays, Human (creature)	Illustration
J_{P3}	Map (Scandina via)	drag input	start interse ction	stop intersect ion	1	1	1	Y	1	n	Boundaries in map	image	replace	national flags	Labeling	Geography, Maps, Flags, Scandinavia, Nations	Map (diagram)
K_{P3}	State of aggregatio n	drag input	start interse ction	stop intersect ion	1	1	1	Y	1	n	Rectangles in back-ground, vertical	image	replace	ice/water/ steam	View state (transforma tion)	Chemistry, Water, Temperature	Photography
L_{P3}	Presentati on media	drag input	start interse ction	stop intersect ion	1	1	1	Y	1	n	Rectangles in back- ground, horizontal	image	replace	chalk board / flip chart / ... / beamer	Progress (transforma tion)	Media, History, Evolution	Photography
M_{P3}	Team	drag input	start interse ction	stop intersect ion	1	1	1	Y	1	n	Bodies	text	replace	names of members	Labeling	Business, Human (creatures)	Photography
N_{P3}	Skeleton (Dragable Label)	drag input	start interse ction	stop intersect ion	1	1	1	Y	1	n	Areas of skeleton parts	text	replace	labels of bones	Labeling	Medicine, Anatomy, Human (creature)	Illustration

Table 2. Discriminative features of interactive graphics.

Checking the interaction attributes in table 2, one can see that the values are invariant (with minor exceptions only) within each set of graphics that implement the same pattern. We shall call them discriminatory dimensions (regarding the interaction). Within the other dimensions one will find more variation in the attribute values. Examples are type of object (e.g. text or image), the shape of the input object (e.g. explicit or implicit boundaries), or the domain (e.g. medicine, geography or technology). These dimensions are considered to be peripheral. There are more dimensions, not captured in the table, that possibly can be identified: target group, number of objects, location and size of objects, affordances, background colour, colour distribution, decorative elements, noise etc. The art of pattern mining is to identify the dimensions that matter, consciously or not.

4.3.1.2 Designing graphics as experimental material

The experimental material consists of graphics that are designed in a way that within each pattern set the graphics are (1) highly invariant on discriminatory dimensions and (2) highly variant in peripheral dimensions. Hence, to see that some graphics belong to the same set, one has to respect the discriminatory and ignore the other dimensions. On the other hand, graphics of different pattern sets are (1) variant on most (but not all) discriminatory dimensions and (2) invariant in at least one peripheral dimension. Hence, graphics that appear to be similar (e.g. same domain or function) are different in some of their interaction attributes. For example, the graphics E_{p1} and J_{p3} both show a map (domain), and flag images (property type) that change to label the map (function). If these dimensions discriminate, one can find a valid pattern set $P_4 = \{ E_{p1} , J_{p3} \}$. The trouble is that E_{p1} and J_{p3} are distinct in event type, activation and deactivation event, number of dragable elements, number of input and output elements. So, the elements of P_4 form a pattern as there are invariant attributes but it is not a pattern of interaction. In opposition, $P2 = \{F_{P2}, G_{P2}, H_{P2}, I_{P2} \}$ forms a pattern of interaction since each of the interaction dimensions is invariant for the elements. Indeed, there are dimensions that distinct elements of P2, for example G_{p2} sets a focus on images whereas H_{p2} translates text. Also, elements of P3 share many interaction attributes with P2. So, one could argue that we merge the two patterns sets into one. Generally for any two graphics g1, g2 $\in$ G, there are invariant and variant attributes and therefore any two graphics could be grouped into a pattern set or not. Because any subset of G is a pattern candidate – depending on the dimension used for discrimination – the set of all possible pattern sets is the power set of G which has the size $P(G)=2^n=2^{14}=16.384$. The proposed patterns P1, P2 and P3 which originally generated the example graphics are only three of these possible 16.384 patterns. The questions is: do other people find the same three pattern sets? Do they find other pattern sets? Are there dominant pattern sets?

4.3.1.3 Participants

The study was run with four groups and a total of 71 participants (17 teachers, 18 teachers, 22 student teachers, and 14 students of media informatics). The participants first got an introduction to the concept of classes. The classification of cars was used as an example, by showing that depending on the point of view two different car types could be within the same class or not. It was carefully explained that no classification should be treated as wrong or less suitable to encourage the participants to come to their own judgements about which interactive graphics were similar. To introduce the concept of interactive graphics, two examples were shown. The first showed the same visual object in different interaction settings. The second showed two different visual objects that used the same interaction form. It was emphasized that the similarity should be judged on the interaction form and not the visual appearance. However, none of the concrete discriminative or peripheral dimensions was mentioned. The introductory examples used different interaction patterns (TRANSPORT OBJECTS, DRAG RESTRICTION) than the actual study. The experiment was divided into 5 phases. Appendix B contains the original worksheets and tasks as they have been handed out to students (in German). The next sections provide an overview of the phases.

4.3.1.4 Phase 1: Pair comparisons

In the first phase, the participants were asked to compare pairs of interactive graphics. In a randomized sequence, which was the same for all participants, two of the 14 interactive graphics appeared in a web browser.

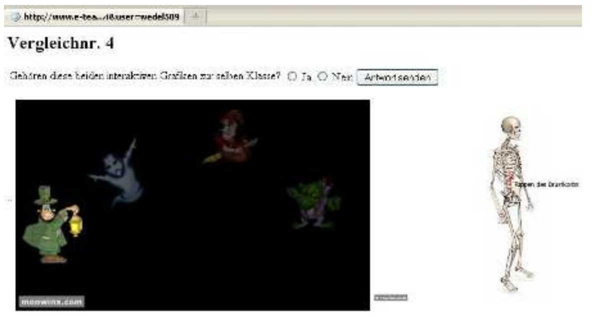

http://www.e-tea.../i8user=wedel509
Vergleichnr. 4
Gehören diese beiden interaktiven Grafiken zur selben Klasse? ○ Ja ○ Nein
Antwort senden
Rippen des Brustkorbs
mouwinx.com

Figure 79. Pair comparisons in browser window.

To compare all entities with each other, a total of 91 comparisons were needed. The first 12 comparisons were repeated at the end to check whether the participants were consistent in their judgements. A time limit of 60 minutes for all 103 comparisons was preset, but no participant needed longer than 30-40 minutes.

4.3.1.5 Phase 2 and 3: Classification

In phase two and phase three (each 10 minutes), the participants were asked to subdivide the 14 graphics into sets of similar graphics. Results of the sorting task had to be noted on paper, while the participants had access to all graphics simultaneously on the computer screen (examples of filled out sheets are found in appendix B). In phase two, subjects were allowed to assign a graphic to more than one set. In phase three, each graphic had to be assigned to one set only. Hence, the set of graphics was divided into disjunctive subsets each representing a pattern.

4.3.1.6 Phase 4 and 5: Naming and describing the classes

In the next phase, the participants were asked to find expressive names for these sub sets that characterized the members. In the fifth and final phase, participants picked up one sub set (of phase three) they felt most confident about. For this sub set - or class, pattern, schema - they had to describe how the interaction works, which elements participate, which functions the elements have, and how the interaction is different to other classes. Participants were also asked to find new examples where the interaction could be applied. The participants had been informed at the beginning about the duration of the experiment and the number of phases but not about the specific tasks in each phase.

The next sections will report for each phase the assumed outcomes (hypotheses), the actual outcomes (falsification or acceptance of hypotheses) and a discussion of the results. Phase one to three can be evaluated quantitative; the evaluation of phase four and five can be supported by qualitative methods of text analysis. In the first three phases we are testing hypotheses that are derived from our theoretical framework. For the independent variables (e.g., properties of interactive graphics, graphics of different assumed interaction patterns, combinations of pairs for comparisons, sequence of presented items, distinction between canonical and non-canonical classification) we can predict and test the quantitative outcome of dependent values such as similarity judgments (frequency of positive/negative judgements), frequency of coherent judgements over time, degree of coherent judgements depending on which classes have been identified, frequency of identified classes and the distance between classifications (quantifying similarities and differences between classes). While quantitative methods are used to test assumed outcomes, qualitative methods can be used to understand and explain why participants have made a decision. It is a way to not only test the predicted outcomes but also explain them. By defining objective criteria we can use a qualitative judgement to score the different descriptions. We can use this scoring for the quantitative analysis about the relation between the quality of a description and the quality of a classification.

4.3.2 Phase 1: Pair comparisons

In the first phase participants were asked to compare pairs of interactive graphics. Since we can assume that at the beginning the judgment is poor because participants are not yet familiar with the set of graphics, the first 12 comparisons were repeated at the end. The sequence of these 12 redundant comparisons was randomized again, i.e. the first and last 12 comparisons involved the same pairs but in a different sequence.

4.3.2.1 Research questions

The patterns we propose are created based on similar and discriminative features. The members of one pattern subset should be homogeneous; the members of two distinct pattern subsets should be inhomogeneous.

Similarity judgments within and between classes

The judgement whether two graphics are similar should depend on whether two graphics exemplify the same pattern. We expect for two graphics that are part of the same pattern sub set that they are judged to be similar. Inversely, for two graphics that are part of distinct pattern sub sets, we would expect the similarity judgement to be negative:

$g1 \in P_x \wedge g2 \in P_y \wedge P_x = P_y$ → positive similarity judgement
$g1 \in P_x \wedge g2 \in P_y \wedge P_x \neq P_y$ → negative similarity judgement

Similarity judgements within super classes

There are different ways of partitioning the graphics into pattern subsets. The two hypothetical pattern subsets P2 and P3 are subparts of another hypothetical set $P_{2+3}=\{F_{P2}, G_{P2}, H_{P2}, I_{P2}, J_{P3}, K_{P3}, L_{P3}, M_{P3}, N_{P3}\}$. If both partitions are equally likely, the judgment will depend on whether one classifies into the higher level subset P_{2+3} or into the two lower level subsets P2 and P3. If two graphics $g2 \in P2$ and $g3 \in P3$ are compared, then g2 and g3 should be dissimilar if one distinguishes between P2 and P3. However, g2 and g3 should be similar if one considers the higher level set P_{2+3}. Both judgements are equally valid. Therefore, we should expect that the pair g2 and g3 is judged to be similar as often as g2 and g3 are judged to be dissimilar.

$g2 \in P2 \wedge g3 \in P3 \wedge P2 \neq P3 \wedge P2 \subset P_{2+3} \wedge P3 \subset P_{2+3}$ → positive and negative judgment occur equally distributed

For example, we assume that SWITCHING STATES is a recognizable pattern. Hence, we expect that the two graphics A_{P1} and $B_{P1}$are judged to be similar. We also assume that ACTIVATOR and ACTIVE AREAS are recognizable patterns. Hence, we expect the two graphics G_{P2} and H_{P2} to be judged as similar. Graphics that exemplify SWITCHING STATES are expected to be judged as different from graphics that exemplify ACTIVATOR or ACTIVE AREAS. Hence, we expect $A_{P1}$and G_{P2} to be judged as dissimilar. If we have one graphic that exemplifies Activator and another graphic that exemplifies ACTIVE AREAS, then we would expect that some participants judge the two graphics as dissimilar, and other participants judge the graphics as similar because they consider them on a higher level of abstraction.

4.3.2.2 Hypotheses about assumed judgements

For each pair of graphics we can propose a hypothesis about the expected outcome. If g1 and g2 are members of a pattern that is clearly distinct to other graphics, then a high percentage of participants should give a positive similarity judgment. If g1 and g2 are members of two clearly distinct patterns then most participants should give a negative similarity judgement. If g1 and g2 are members of two patterns that are not clearly distinct, i.e. the two patterns could be seen as a more general pattern, then about one half of the participants should give a positive similarity judgement and the other half gives a negative judgement. Taking these assumptions into account we can propose a hypothesis for the outcome of each pair comparison. Let similar(g1,g2) be a function that returns the percentage of participants that have judged the pair of graphics g1 and g2 to be similar. Then we can assume the following hypotheses:

H1.1: $g1 \in P_x \wedge g2 \in P_y \wedge P_x \neq P_y \rightarrow$ similar (g1, g2) < 33,33%
H1.2: $g1 \in P_x \wedge g2 \in P_y \wedge P_x \subset P_z \wedge P_y \subset P_z \wedge P_z = P_x \dot{\cup} P_y \rightarrow$ 33,33% < similar (g1,g2) < 66,66%
H1.3: $g1 \in P_x \wedge g2 \in P_y \wedge P_x = P_y \rightarrow$ similar (g1,g2) > 66,66%

Each hypothesis predicts that the outcome will be found within a specific range. The range is either at the bottom (H1.1 for graphics from distinct patterns), in the middle (H1.2 for graphics that can be judged either to be or not to be part of distinct patterns), or at the top (H1.3 for graphic of the same pattern). Each range reflects a tendency. To ensure that similar(g1, g2) does not fall into one of the ranges by chance we check whether the confidence interval of similar(g1,g2) fully falls into the predicted range. We use a confidence interval where alpha=0,01. Since each hypothesis spans a wide range, we require the location of the confidence interval to be completely within the predicted range. If similar(g1,g2) returns a mean value outside the expected range, we directly reject the hypothesis without considering the confidence interval.

4.3.2.3 Results

Table 3 shows for each pair of graphics the expected outcome (which hypothesis was assigned), the actual outcome in each group (which percentage of participants of one group judged the pair to be similar) and the outcome of the whole group (which percentage of all participants from the four groups judged the pairs to be similar). Between the groups there are only a few differences whether the mean is located in the hypothesized range. The cases where the outcome of one group is different to the outcome of all groups are highlighted in the table. The column "significance" refers to the outcome of the whole group and means the following: false (f) means that the mean is outside the expected range; not significant (n.s.) means that the mean is within the expected range but the confidence interval is not fully in that range; significant (*) means the confidence interval for alpha=0.05 is within the range; very significant (**) means the confidence interval for alpha=0,01 is within the predicted range.

#	Interaction type	Expected outcome for mean	First group	Second Group	Third Group	Fourth Group	All Groups	Significance
1	Different class	0-33%	18,18%	17,65%	50,00%	7,14%	23,94%	n.s
2	Same class	68-100%	36,36%	23,53%	33,33%	14,29%	28,17%	F
3	Super/ /Sub class	33-68%	54,55%	70,59%	66,67%	28,57%	56,34%	n.s.
4	Super/Sub class	33-68%	72,73%	70,59%	66,67%	64,29%	69,01% !!!	f
5	Same class	68-100%	40,91%	64,71%	50,00%	64,29%	53,52%	f
6	Different class	0-33%	59,09%	47,06%	72,22%	28,57%	53,52%	f
7	Different class	0-33%	13,64%	35,29%	22,22%	7,14%	19,72%	*
8	Different class	0-33%	40,91%	29,41%	44,44%	35,71%	38,03%	f
9	Different class	0-33%	0,00%	5,88%	5,56%	14,29%	5,63%	**
10	Different class	0-33%	22,73%	23,53%	33,33%	28,57%	26,76%	n.s
11	Super/Sub class	33-68%	54,55%	70,59%	55,56%	35,71%	54,93%	n.s.
12	Same class	68-100%	86,36%	82,35%	77,78%	78,57%	81,69%	*
13	Different class	0-33%	9,09%	11,76%	16,67%	0,00%	9,86%	**
14	Same class	68-100%	95,45%	94,12%	83,33%	92,86%	91,55%	**
15	Different class	0-33%	22,73%	35,29%	27,78%	0,00%	22,54%	n.s
16	Different class	0-33%	9,09%	11,76%	27,78%	21,43%	16,90%	**
17	Different class	0-33%	68,18%	88,24%	77,78%	78,57%	77,46%	f
18	Different class	0-33%	0,00%	5,88%	5,56%	0,00%	2,82%	**
19	Different class	0-33%	22,73%	17,65%	38,89%	35,71%	28,17%	n.s
20	Super/Sub class	33-68%	45,45%	70,59%	50,00%	78,57%	59,15%	n.s
21	Different class	0-33%	13,64%	5,88%	11,11%	14,29%	11,27%	**
22	Different class	0-33%	68,18%	94,12%	77,78%	85,71%	80,28%	F
23	Super/Sub class	33-68%	59,09%	64,71%	77,78%	64,29%	66,20%	n.s
24	Different class	0-33%	27,27%	35,29%	33,33%	7,14%	26,76%	n.s
25	Same class	68-100%	100,00%	76,47%	100,0%	100,00%	94,37%	**
26	Super/Sub class	33-68%	45,45%	70,59%	22,22%	42,86%	45,07%	n.s.
27	Same class	68-100%	95,45%	88,24%	88,89%	92,86%	91,55%	**
28	Super/Sub class	33-68%	31,82%	58,82%	50,00%	35,71%	43,66%	n.s
29	Same class	68-100%	90,91%	70,59%	72,22%	92,86%	81,69%	*
30	Different class	0-33%	4,55%	17,65%	11,11%	7,14%	9,86%	**
31	Different class	0-33%	4,55%	11,76%	16,67%	0,00%	8,45%	**
32	Super/Sub class	33-68%	45,45%	52,94%	61,11%	35,71%	49,30%	**
33	Super/Sub class	33-68%	63,64%	64,71%	83,33%	71,43%	70,42%	f
34	Different class	0-33%	0,00%	17,65%	11,11%	14,29%	9,86%	**
35	Different class	0-33%	4,55%	23,53%	11,11%	7,14%	11,27%	**
36	Different class	0-33%	9,09%	11,76%	22,22%	7,14%	12,68%	**
37	Same class	68-100%	13,64%	47,06%	44,44%	21,43%	30,99%	f
38	Different class	0-33%	0,00%	11,76%	16,67%	14,29%	9,86%	**
39	Different class	0-33%	0,00%	5,88%	5,56%	7,14%	4,23%	**
40	Different class	0-33%	13,64%	17,65%	27,78%	14,29%	18,31%	**
41	Different class	0-33%	4,55%	17,65%	16,67%	0,00%	9,86%	**
42	Same class	68-100%	4,55%	35,29%	38,89%	14,29%	22,54%	f
43	Same class	68-100%	100,00%	94,12%	83,33%	85,71%	91,55%	**

#	Interaction type	Expected outcome for mean	First group	Second Group	Third Group	Fourth Group	All Groups	Significance
44	Super/Sub class	33-68%	63,64%	70,59%	61,11%	57,14%	63,38%	n.s
45	Same class	68-100%	90,91%	88,24%	94,44%	92,86%	91,55%	**
46	Different class	0-33%	0,00%	29,41%	11,11%	7,14%	11,27%	**
47	Super/Sub class	33-68%	50,00%	70,59%	61,11%	50,00%	57,75%	n.s.
48	Different class	0-33%	0,00%	5,88%	0,00%	14,29%	4,23%	**
49	Super/Sub class	33-68%	59,09%	64,71%	55,56%	64,29%	60,56%	n.s
50	Different class	0-33%	4,55%	11,76%	5,56%	7,14%	7,04%	**
51	Same class	68-100%	90,91%	82,35%	72,22%	78,57%	81,69%	*
52	Different class	0-33%	0,00%	5,88%	0,00%	7,14%	2,82%	**
53	Different class	0-33%	0,00%	17,65%	0,00%	7,14%	5,63%	**
54	Different class	0-33%	13,64%	5,88%	0,00%	0,00%	5,63%	**
55	Same class	68-100%	0,00%	5,88%	22,22%	0,00%	7,04%	f
56	Different class	0-33%	0,00%	17,65%	0,00%	7,14%	5,63%	**
57	Different class	0-33%	0,00%	23,53%	27,78%	42,86%	21,13%	*
58	Different class	0-33%	0,00%	0,00%	0,00%	7,14%	1,41%	**
59	Super/Sub class	33-68%	54,55%	70,59%	61,11%	50,00%	59,15%	n.s
60	Same class	68-100%	81,82%	88,24%	83,33%	85,71%	84,51%	**
61	Super/Sub class	33-68%	50,00%	52,94%	50,00%	42,86%	49,30%	**
62	Different class	0-33%	0,00%	0,00%	5,56%	0,00%	1,41%	**
63	Different class	0-33%	0,00%	17,65%	11,11%	7,14%	8,45%	**
64	Same class	68-100%	90,91%	88,24%	72,22%	71,43%	81,69%	*
65	Different class	0-33%	0,00%	11,76%	16,67%	28,57%	12,68%	**
66	Same class	68-100%	100,00%	76,47%	72,22%	78,57%	83,10%	**
67	Different class	0-33%	4,55%	5,88%	27,78%	0,00%	9,86%	**
68	Different class	0-33%	22,73%	17,65%	27,78%	7,14%	19,72%	*
69	Different class	0-33%	13,64%	11,76%	11,11%	7,14%	11,27%	**
70	Different class	0-33%	4,55%	29,41%	11,11%	14,29%	14,08%	**
71	Same class	68-100%	0,00%	17,65%	22,22%	7,14%	11,27%	f
72	Same class	68-100%	81,82%	76,47%	72,22%	64,29%	74,65%	n.s
73	Different class	0-33%	4,55%	23,53%	22,22%	7,14%	14,08%	**
74	Super/Sub class	33-68%	54,55%	58,82%	77,78%	71,43%	64,79%	n.s
75	Same class	68-100%	90,91%	76,47%	55,56%	64,29%	73,24%	n.s
76	Super/Sub class	33-68%	59,09%	70,59%	66,67%	64,29%	64,79%	n.s
77	Different class	0-33%	4,55%	5,88%	11,11%	7,14%	7,04%	**
78	Same class	68-100%	100,00%	88,24%	83,33%	85,71%	90,14%	**
79	Same class	68-100%	68,18%	70,59%	50,00%	64,29%	63,38%	f
80	Same class	68-100%	90,91%	88,24%	66,67%	71,43%	80,28%	*
81	Different class	0-33%	18,18%	29,41%	27,78%	28,57%	25,35%	n.s
82	Different class	0-33%	0,00%	5,88%	22,22%	0,00%	7,04%	**
83	Different class	0-33%	9,09%	23,53%	22,22%	7,14%	15,49%	**
84	Same class	68-100%	90,91%	88,24%	72,22%	85,71%	84,51%	**
85	Different class	0-33%	4,55%	23,53%	11,11%	7,14%	11,27%	**
86	Super/Sub class	33-68%	59,09%	76,47%	83,33%	78,57%	73,24%	f
87	Same class	68-100%	95,45%	82,35%	72,22%	71,43%	81,69%	*
88	Same class	68-100%	0,00%	23,53%	11,11%	14,29%	11,27%	f
89	Same class	68-100%	100,00%	100,00%	100,0%	100,00%	100,00%	**
90	Different class	0-33%	0,00%	17,65%	0,00%	0,00%	4,23%	**
91	Super/Sub class	33-68%	54,55%	70,59%	77,78%	50,00%	63,38%	n.s
92/3r	Super/Sub class	33-68%	31,82%	64,71%	55,56%	50,00%	49,30%	**
93/7r	Different class	0-33%	0,00%	17,65%	5,56%	7,14%	7,04%	**
94/10r	Different class	0-33%	0,00%	23,53%	5,56%	14,29%	9,86%	**
95/6r	Different class	0-33%	9,09%	35,29%	27,78%	21,43%	22,54%	n.s
96/12r	Same class	68-100%	95,45%	88,24%	83,33%	78,57%	87,32%	**
97/2r	Same class	68-100%	4,55%	23,53%	16,67%	7,14%	12,68%	f

#	Interaction type	Expected outcome for mean	First group	Second Group	Third Group	Fourth Group	All Groups	Significance
98/4r	Super/Sub class	33-68%	50,00%	64,71%	72,22%	50,00%	59,15%	n.s
99/11r	Super/Sub class	33-68%	40,91%	52,94%	44,44%	42,86%	45,07%	n.s
100/5r	Same class	68-100%	86,36%	76,47%	72,22%	78,57%	78,87%	n.s
101/9r	Different class	0-33%	0,00%	5,88%	0,00%	0,00%	1,41%	**
102/1r	Different class	0-33%	4,55%	11,76%	5,56%	14,29%	8,45%	**
103/8r	Different class	0-33%	4,55%	23,53%	16,67%	21,43%	15,49%	**

Table 3. Results of pair comparisons.
Different class -> mean expected between 0-33%; Super or sub class possible -> mean expected between 33-68%; Same class -> mean expected between 68-100%; Green text: mean as expected; Red text: mean not as expected
Orange cells # 1-12: new graphic introduced , Blue cells # 92-103: repetition of first 12 comparisons in different order (92/r3 means #92 is a repetition of #3); Gray cells: indicates different outcome for this group

Table 3 shows all pair comparisions in the randomized order that was used in the experiment. It also shows differences between the four groups. Table 4 shows the results of all groups in a crosstabulation (same values as in row "all groups" of table 3). Each cell contains the expected value range, the pair comparison index, and the actual mean (green: as expected, red: not as expected).

	A_{P1} Flower	B_{P1} Camera	C_{P1} Skeleton (Single labels)	D_{P1} Com-puter Board	E_{P1} Map (UK)	F_{P2} Night - watchman	G_{P2} Animals	H_{P2} Voca-bulary	I_{P2} X-Rays	J_{P3} Map (Scan-dinavia)	K_{P3} State of aggre-gation	L_{P3} Presen-tation media	M_{P3} Team	N_{P3} Skeleton (Drag-able Label)
A_{P1} Flower	-	(>67%) # 84: 84,51%	(>67%) # 79: 63,38%	(>67%) # 88: 11,27%	(>67%) # 2: 28,17% # 97: 12,68%	(<34%) # 46: 11,27%	(<34%) # 36: 12,68%	(<34%) # 13: 9,86%	(<34%) # 41: 9,86%	(<34%) # 77 7,04%	(<34%) # 63: 8,45%	(<34%) # 56: 5,63%	(<34%) # 16: 16,90%	(<34%) # 35: 11,27%
B_{P1} Camera	-	-	(>67%) # 60: 84,51%	(>67%) # 71: 11,27%	(>67%) # 55: 7,04%	(<34%) # 30: 9,86%	(<34%) # 18: 2,82%	(<34%) # 53: 5,63%	(<34%) # 90: 4,23%	(<34%) # 52 2,82%	(<34%) # 58: 1,41%	(<34%) # 9: 5,63% # 101: 1,41%	(<34%) # 67: 9,86%	(<34%) # 38: 9,86%
C_{P1} Skeleton (Single labels)	-	-	-	(>67%) # 37: 30,99%	(>67%) # 42: 22,54%	(<34%) # 48: 4,23%	(<34%) # 31: 8,45%	(<34%) # 10: 26,76% # 94: 9,86%	(<34%) # 65: 12,68%	(<34%) # 7: 19,72% # 93: 7,04%	(<34%) # 39: 4,23%	(<34%) # 34: 9,86%	(<34%) # 40: 18,31%	(<34%) # 57: 21,13%
D_{P1} Computer Board	-	-	-	-	(>67%) # 14: 91,55%	(<34%) # 85: 11,27%	(<34%) # 24: 26,76%	(<34%) # 15: 22,54%	(<34%) # 69: 11,27%	(<34%) # 68: 19,72%	(<34%) # 62: 1,41%	(<34%) # 50: 7,04%	(<34%) # 6: 53,52% #95: 22,54%	(<34%) # 81: 25,35%
E_{P1} Map (UK)	-	-	-	-	-	(<34%) # 1: 23,94% # 102: 8,45%	(<34%) # 73: 14,08%	(<34%) # 8: 38,03% #103 : 15,49%	(<34%) # 21: 11,27%	(<34%) # 19: 28,17%	(<34%) # 54: 5,63%	(<34%) # 82: 7,04%	(<34%) # 70: 14,08%	(>34%) # 83 15,49%
F_{P2} Night watchman	-	-	-	-	-	-	(>67%) # 29: 81,69%	(>67%) # 64: 81,69%	(>67%) # 78: 90,14%	(33-68) # 49: 60,56%	(33-68) # 28: 43,66%	(33-68) # 20: 59,15%	(33-68) # 74: 64,79%	(33-68) # 4: 69,01% # 98: 59,15%
G_{P2} Animals	-	-	-	-	-	-	-	(>67%) # 27: 91,55%	(>67%) # 66: 83,10%	(<34%) # 22: 80,28%	(33-68) # 11: 54,93% #99: 45,07%	(33-68) # 26: 45,07%	(33-68) # 86: 73,24%	(33-68) #91: 63,38%
H_{P2} Vocabulary	-	-	-	-	-	-	-	-	(>67%) # 12: 81,69% # 96: 87,32%	(<34%) # 17: 77,46%	(33-68) # 3: 56,34% # 92: 49,30%	(33-68) # 32: 49,30%	(33-68) # 76: 64,79%	(33-68) # 33: 70,42%
I_{P2} X-Rays	-	-	-	-	-	-	-	-	-	(33-68) # 44: 63,38%	(33-68) # 61: 49,30%	(33-68) # 59: 59,15%	(33-68) # 47: 57,75%	(33-68) # 23: 66,20%
J_{P3} Map Scandinavia	-	-	-	-	-	-	-	-	-	-	(>67%) # 51: 81,69%	(>67%) # 87: 81,69%	(>67%) # 45: 91,55%	(>67%) # 25: 94,37%
K_{P3} State of aggregation	-	-	-	-	-	-	-	-	-	-	-	(>67%) # 43: 91,55%	(>67%) # 75: 73,24%	(>67%) # 80: 80,28%
L_{P3} Presentation media	-	-	-	-	-	-	-	-	-	-	-	-	(>67%) # 72: 74,65%	(>67%) # 5: 53,52% # 100: 78,87%
M_{P3} Team	-	-	-	-	-	-	-	-	-	-	-	-	-	(>67%) #89: 100,00 %
N_{P3} Skeleton (Dragable Label)	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4. Results of pair comparisons (means of all groups) - crosstabulation.

4.3.2.4 Discussion

We will first focus on the hypotheses that members of our hypothetical patterns are judged to be similar in pair comparison are verified. Since there is not much variation between the groups we will consider only the results for the whole group (participants of all four groups).

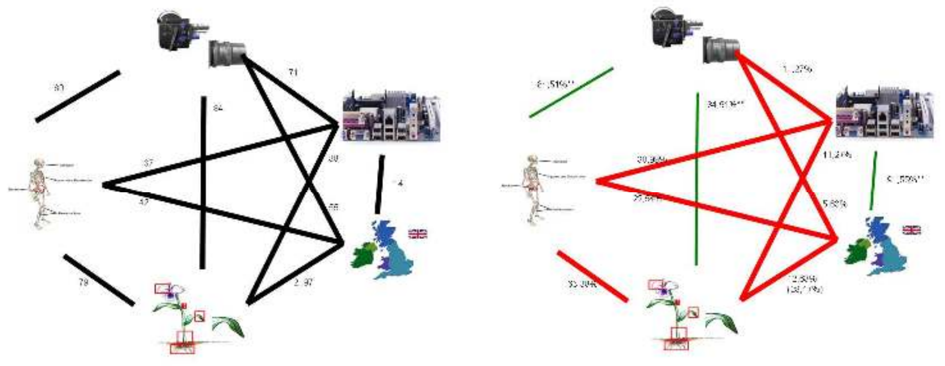


Figure 80. Switching States - Left: Assumed high similarity judgement for pair comparison # x – Right: Actual similarity judgement

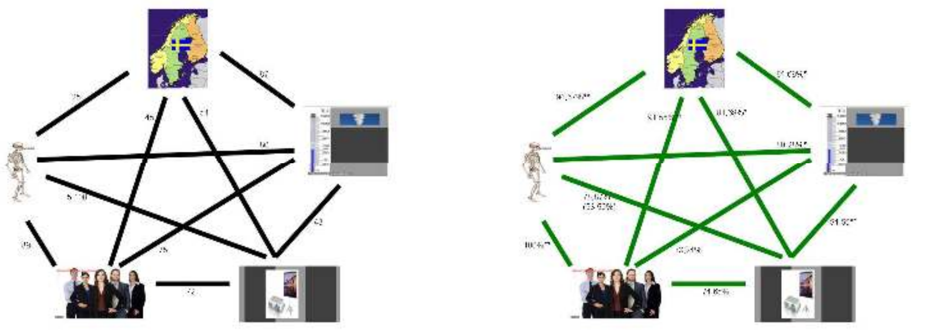


Figure 81. Active Areas - Left: Assumed high similarity judgement for pair comparison # x – Right: Actual similarity judgement

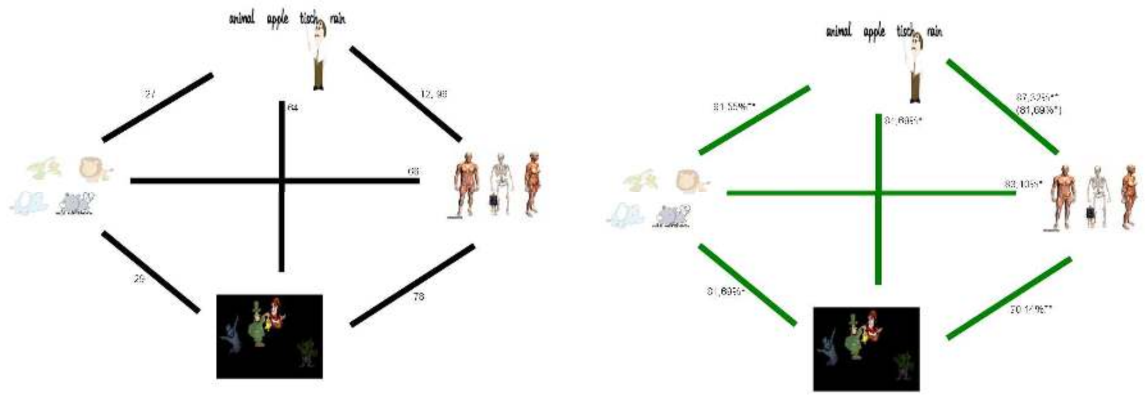


Figure 82. Activator - Left: Assumed high similarity judgement for pair comparison # x – Right: Actual similarity judgement

The members of ACTIVATOR and ACTIVE AREAS are indeed all judged as similar in pair comparisons. For SWITCHING STATES, however, not all members are perceived as similar. There is a similarity between the graphics that are triggered by mouse-rollovers. The sub patterns ROLLOVER and ON/OFF BUTTON are distinguished by the participants. However, while they judge the camera B_{P1} similar to both the flower A_{P1} and the clickable skeleton C_{P1}, they did not judge A_{P1} and C_{P1} as similar. There is an important difference between the two: the flower (A_{P1}) allows only one element to be switched ON whereas the skeleton (C_{P1}) allows multiple labels to be switched ON. Thus, A_{P1} consists of radio buttons and C_{P1} consists of simple, independent buttons. Yet the camera is judged to be similar to both. This is reasonable if we consider the interaction: it is similar to the clickable skeleton in that it provides multiple elements that can be switched on; it is similar to the camera in that it shows enlarged pictures of subsections. B_{P1} shares different similarities with A_{P1} than with C_{P1}.

The implication of these pair comparison is that participants perceive rollover and clickable buttons as clearly distinct. Simple buttons and radio buttons are also perceived as distinct interactions. However, if further similarities are shared, such as the output effect, radio and simple buttons may be perceived as similar interactions.

We can also see that graphics from disjunctive patterns produce significant outcomes. However, for hypotheses that follow H1.2 (different possible classifications) we find many outcomes that are not significant. While the mean is within the specified range, the confidence interval is not.

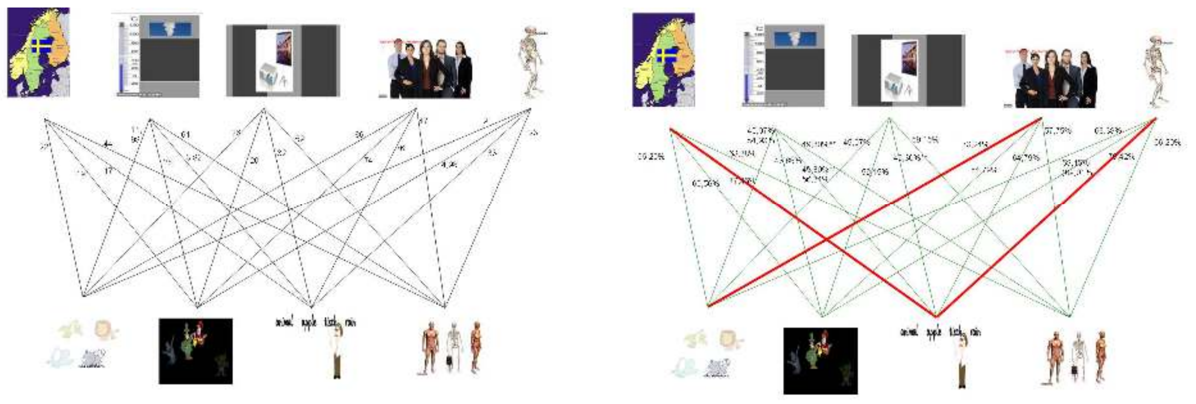


Figure 83. Active Areas and Activator - Left: Assumed medium similarity judgement for pair comparison # x – Right: Actual similarity judgement

The reason why we can be less confident about these outcomes is that there is a higher variance in the judgements. This can be seen if we compare the means of the different groups. Where the confidence interval of the mean of the whole group is not fully within the range, we can see that for some groups the mean is in the expected range and for others it is not. The deviation from the mean is much higher for graphics that can either belong to the same super class P2+3 or to the distinct classes P2 and P3. For those graphics we know that there are at least some discriminatory features – otherwise the specialization into P2 and P3 would not be justified. But the differences can be accompanied by further variations or similarities on peripheral features. If more variations add to the interaction difference, then it is more likely that two graphics are judged to be different. Likewise, if there are many similarities on other features, it is more likely that graphics are judged to be similar. The variance of similarity judgements shows that P2+3 is more heterogeneous than P2 and P3.

If we consider the table of outcomes we will see that at the beginning there are more hypotheses falsified. This is to be expected since at the beginning participants are not yet familiar with the graphics. Once participants have seen all of the graphics they build up their own schemas of interaction. At the end of the phase, participants have more stable schemas. If two graphics match the same schema they will be judged as similar. This is evident in the repetition of the first 12 pair comparisons at the end.

#	Interaction type	Compared graphic	First run at begin	Significance	#	Second run at end	Significance
1	Different class	$F_{P2} – E_{P1}$	23,94%	n.s	102	8,45%	**
2	Same class	$E_{P1} – A_{P1}$	28,17%	f	97	12,68%	f
3	Super/ /Sub class	$K_{P3} – H_{P2}$	56,34%	*	92	49,30%	**
4	Super/Sub class	$F_{P2} – N_{P3}$	69,01%	n.s	98	59,15%	n.s
5	Same class	$L_{P3} – N_{P3}$	53,52%	f	100	78,87%	*
6	Different class	$D_{P1} – M_{P3}$	53,52%	f	95	22,54%	**
7	Different class	$C_{P1} – J_{P3}$	19,72%	n.s	93	7,04%	**
8	Different class	$H_{P2} – E_{P1}$	38,03%	f	103	15,49%	**
9	Different class	$K_{P3} – B_{P1}$	5,63%	**	101	1,41%	n.s
10	Different class	$H_{P2} – N_{P3}$	26,76%	n.s	94	9,86%	**
11	Super/Sub class	$K_{P3} – G_{P2}$	54,93%	*	99	45,07%	n.s.
12	Same class	$H_{P2} – I_{P2}$	81,69%	*	96	87,32%	**

Table 5. Pair comparisons at the beginning and at the end.

While in the first 12 comparisons many hypotheses have been falsified the last 12 comparisons (which consist of the same pairs) showed more confirmations. There is only one hypothetical outcome that is falsified and three positive outcomes that are not significant. The other 9 outcomes support the hypotheses and are either significant or very significant.

4.3.3 Phase 2: Classification with overlapping classes

In this phase, all participants were asked to circle graphics that belong to one class. To introduce the concept of classification, a number of static images were shown on a slide. The images, then, were put into several classes and some of the images were part of multiple classes.

We asked the participants to find classes based on similarities of the interactive graphics they just had seen. At the computer screen they had access to all 14 interactive graphics at the same time. On a paper sheet that showed the 14 graphics, participants were allowed to circle as many classes as they liked. Graphics were allowed to be members of several classes.

4.3.3.1 Research questions and hypotheses

Since the classification of the graphics should be based on similarity we assumed that participants would find both the hypothetical patterns P1, P2, and P3, as well as other patterns – because different criteria could be used for classification. Hence, our hypotheses:

> **H2.1:** The hypothetical patterns P1, P2, and P3 should be found most frequently. At least one of them should have the highest frequency of all found patterns.
> **H2.2:** There should be other patterns Px that occur roughly as often as P1, P2 or P3. Px should occur more frequently than one of P1, P2, or P3.

We are confident that P1, P2 and P3 are good classifications and therefore will be identified most frequently (H2.1.). However, we do not assume that they are the only good classifications. If the graphics are considered from a different perspective, other patterns may emerge. Note that in this particular phase of the experiment we permit participants to take as many perspectives as they wish. Each graphic is allowed to be included in several classes, including sub or super classes and using alternative criteria. Therefore, we assume that there is at least one other class that is as good as P1, P2 or P3 (H.2.2).

4.3.3.2 Results

In total, 377 sets have been marked (an average of 5,31 sets per participant). These 377 sets consist of only 117 different sets, meaning that some sets have been frequently used[29]. Table 6 shows the distribution of pattern sets.

Number of sets	Found by n participants	% partipants who found the set	Set index	Patterns
1	43	61,43%	P_1=P1_1	Switching States (Rollover)
1	42	60,00%	P_2=P1_2	Switching States (Clicks)
1	23	32,86%	P_3=P2	Activator
1	22	31,43%	P_4=P2 + P3	Union of Active Area and Activator
1	18	25,71%	P_5=P1_{1_1}	
1	17	24,29%	P_6=P3	Active Areas
1	14	20,00%	P_7=P3_1	
1	13	18,57%	P_8 =P4	
1	10	14,29%	P_9	
1	9	12,86%	P_{10}	
1	8	11,43%	P_{11}	
1	6	8,57%	P_{12}	
3	5	7,14%	$P_{13..15}$	
3	4	5,71%	$P_{16..18}$	
5	3	4,29%	$P_{19..23}$	
16	2	2,86%	$P_{24..39}$	
78	1	1,43%	$P_{40..117}$	

Table 6. Distribution of identified sets

[29] One participant sheet was not properly filled out, i.e. different interpretations of the marked sets were possible. Therefore the analysis is based on 70 participants.

The pattern P1 = {A_{P1}, B_{P1}, C_{P1}, D_{P1}, E_{P1} } was not found a single time; however its two sub patterns $P1_1$= {A_{P1}, B_{P1}, C_{P1}} and $P1_2$= , {D_{P1}, E_{P1} } were each found by more than 60% of the participants. The patterns P2 (32,86%) and P3 (24,29%) both were found frequently. Hence, we can accept the proposed patterns P2 and P3. However, we have to reject the hypothesis that P1 is at an appropriate level of abstraction. The participants only found the more specific sub patterns $P1_1$ and $P1_2$.

The distribution table also shows that there are two other patterns that occur at least as often as one of the proposed patterns. The pattern P_{2+3} is the super class of P2 and P3. It was found by 22 (31,43%) participants – more frequently than P3. There is another class that was found more frequently than P3. The class $P1_{1_1}$= {A_{P1}, B_{P1}} is a specialization of $P1_1$ because it contains only the members that show a zoom effect. $P1_{1_1}$ has been found by 18 participants (25,71%). That means we have at least two patterns that are found as frequently as one of the proposed patterns. Pattern $P3_1$ = {K_{P1}, L_{P1}} is a sub pattern of P3 that only includes graphics where the moving object is an image (instead of text) and the background only consists of boxes (instead of another image). It has been found 14 times (20%).

Pattern P4 = {E_{P1}, J_{P3}} is not a sub pattern of any of the proposed patterns. Rather it is based on the fact that both members show maps and flags of the countries. 13 participants (18,57%) have found that pattern.

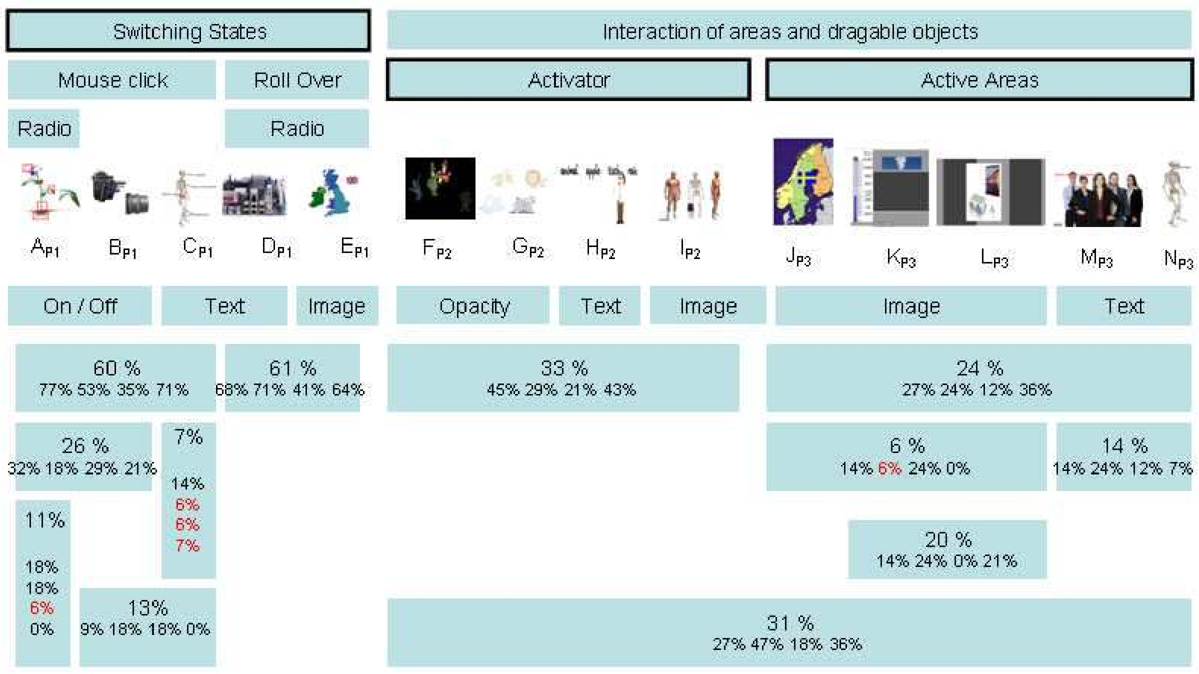


Figure 84. Identified sets (overlapping sets)

Figure 84 shows that most sets found were similar to the sets we expected. The graphic reads as follows: the graphics shows the examples G={A_{P1}, B_{P1}, C_{P1}, D_{P1}, E_{P1}, F_{P2}, G_{P2}, H_{P2}, $I_{P,}$ J_{P1}, K_{P1}, L_{P1}, M_{P1}, N_{P1}}. Above the graphics we see boxes, each indicating a subset expected to be found, the hypothetical sets. Bold boxes indicate that the set was expected to be dominant, e.g. SWITCHING STATE is bold but there are also its sub patterns ON/OFF, ROLLOVER and RADIO. Each of the boxes underneath the graphics represents a set that was found multiple times (at least three times) by the participants, the empirical sets. A box spans horizontally over all graphics the participants had included in that set. The large percentage value shows how many participants from the three groups found that set. The smaller percentage values show the findings of each single group. For example, the set with ACTIVE AREA graphics was found by 24% of all participants, more specific by 27% in group 1, by 24% in group 2, by 12% in group 3, and by 36% in group 4. The sets found were equal to, subsets or unions of the proposed patterns. In some cases the subsets have been found by only one person of each group. In such cases the percentage value is also provided nevertheless but a red font indicates that the percentage means only one person.

4.3.3.3 Discussion

Given that there are 16.384 possible classes and participants defined 377 classes, we can assume that the same class could be found coincidently by two members. However, the probability that the same class is found coincidently by three participants is $p<0,04$ (significant) and the probability that the same class is found coincidently by four participants is p<0,0003 (very significant). The probability for higher frequencies is extremely low (see appendix C). Hence, we can be highly confident that the same classes were found not by chance but because these classes are more meaningful. That the same classes were found in each of the different groups also supports this confidence.

Since participants were allowed to find overlapping classes, they were able to externally represent any schema they might have induced. This included schemas not only about the interaction but also about applications in educational contexts (geometry, or medicine).

While participants frequently found the super class of our proposed class P2 and P3 they never found the class P1. This is remarkable because from a technological point of view the differences between the sub patterns of P1 are much fewer than between P2 and P3. Instead “Switching States” (P1) was subdivided into "Mouse click" and "Rollover". Some participants sub-divided "Mouse click" further, however, in different ways. Some took the
the property type (on/off or text) as the splitting criteria while others took the number of simultanously activated elements (radio button) as the splitting criteria.

4.3.4 Coherence between classification and pair comparisons

If we assume that schemas are induced based on previous experience, then we should expect that at the beginning of phase 1 the schemas are not developed and at the end of phase 1 the schemas are strong enough to perform the classification of phase 2.

4.3.4.1 Research questions

For each individual participant, the pair comparisons of phase 1 should be coherent with her or his own schema. Hence, if one participant judged g1 and g2 to be similar, then there must be at least one class that has g1 and g2 as members. This does not exclude that there are other classes that have only one of the two graphics as a member. Classes could be created based on different criteria. g1 and g2 can be similar in respect to the triggering event but distinct in respect to the properties that change. g1 and g2 are not required to always show up in conjunction. However, if g1 and g2 are judged to be similar in the pair comparisons there should be at least one class that does have both graphics as members.

On the other hand, if g1 and g2 are not judged to be similar in some respect during the pair comparison, then there should be not a single class that has g1 and g2 as members. For two graphics to be in the same class requires that there are at least some meaningful similarities. If a participant judges two graphics to be different but puts them into one class that would be incoherent.

Definition of coherence between pair comparisons and classification

A pair comparison of g1 and g2 is coherent to one’s individual classification if

- g1 and g2 are judged to be similar (phase 1) and there is at least one class that has g1 and g2 as members
- g1 and g2 are judged to be dissimilar and there is no class that has g1 and g2 as members.

For each participant we can track the number of coherent pair comparisons. The percentage of coherent judgements in relation to the number of performed pair comparisons should increase over time because the schemas equilibrate. If 0% of the pair comparisons are coherent then a participant always judges incoherently to her or his own classes. If 100% of the pair comparisons are coherent, then a participant always judges coherently. If only 50% of the pair comparisons are coherent the judgements could as well be random.

Measurement of judgement performance

To measure the performance of judgements, let coherence(a,b) be a function that returns the mean of coherent judgements of all participants, where a and b define the range of pair comparisons considered. For example, coherence (1, 12) considers the first 12 pair comparisons of all participants and returns the relative number of comparisons that are coherent to one’s individual classification. If person A makes 8 coherent judgements and person B makes 10 coherent judgements we will have on average 9 coherent judgements (75% coherent judgments for 12 comparisons). Note that person A and person B both can judge two graphics as similar but for person A this means a coherent judgement and for person B this means an incoherent judgment. This happens if person A has later identified a class that includes the two graphics whereas person B has not.

4.3.4.2 Hypotheses about the development of judgment performance

From these considerations we can derive the following hypotheses:

1. The first pair comparisons are not based on prior experience. Hence, a participant has not build up any schemas about the interactive graphics. Coherence to later built schemas is nearly random, i.e. for the first comparisons the percentage of coherent decisions should be around 50%. Since the first 12 pairs include at least one unknown graphic, we can assume that on average for each of the first 12 comparisons only 40-60% are coherent with the schemas later found.

   H2.3: 40% < coherence (1,12) < 60%

2. Since schemas build up over time, we can assume that pair comparisons become more and more coherent with the classes of an individual participant. The trend for each participant is an increasing percentage of coherent judgements. We can consider the mean performance of coherent judges for each of the four quartiles. We assume that the performance of the fourth quartile is better than the third which is better than the second which is better than the first.

   H2.4.:
   coherence (1,26) < coherence (27 , 52) < coherence (53, 78) < coherence(79, 103)

3. At the end of phase 1 we can expect that the participants have built their own schemas. Hence, the last pair comparisons should be highly coherent to the schemas found. In particular, the last 12 comparisons (which include the same pair comparisons as the first 12) should be highly coherent. On average around 80%-100% should be judged correctly.

   H2.5: 80% < coherence (92,103) < 100%

To test H2.3 and H2.5 we can check whether the confidence interval of the mean of coherent judgments is within the predicted ranges. The ranges are estimations. It may be that participants start better right at the beginning (due to obvious similarities in the graphics) or perform not as good as expected. We would accept H2.3 and H2.5 if the outcome is significant (p<0,05). For H2.4.2 we can run paired t-tests to see whether a growth of the mean is significant. Since we compare the performance of the same participants directly (a within-subject design) we can use paired t-tests to see whether a found growth is significant. Since we assume that the schemas are induced while the graphics are seen several times, we are very confident to find the growth predicted in H2.4.2. Hence we expect very significant outcomes (p<0,01) in order to accept the hypotheses.

4.3.4.3 Results

Table 7 shows the mean, standard deviation, and confidence interval for H2.3 and H2.5. H2.3 and H2.5 both need to be rejected because their confidence intervals are not within the predicted range. At the beginning the participants perform better than expected (better than random). The performance at the end is not as good as expected. Even at the end of phase 1 about 25% of the judgments are not coherent.

	Mean	SD	Confidence interval min	Confidence interval max	Significance
First 12 comparisons	65,84%	0,17366	61,81%	69,88%	reject
Last 12 comparisons	78,98%	0,13631	75,81%	82,15%	reject

Table 7. Performance of first and last twelve comparisons

The performance increases over time as predicted. If we compare each of the four quartiles we can accept H2.4 as table 8 shows. The difference between the means of each quartile is very significant.

	1. Quartile	2. Quartile	3. Quartile	4.Quartile
Mean	66,98%	71,85%	76,88%	78,39%
Standard Error	0,02791586	0,01784974	0,01613306	0,01783453
P(T<=t) one-tail		1,6632E-05 **	1,395E-16 **	7,1732E-06 **

Table 8. Performance increase over quartiles.

4.3.4.4 Discussion

The confidence interval for H2.3 suggests that participants perform better than 50% on average already on the first 12 comparisons. That means that the participants quickly recognize similarities between the interactive graphics. This is also supported if we take a look at the mean for each of the first 12 comparisons. It it is always between 60-76% (see appendix A).

The growth of the mean performance is very significant for each quartile. That the error probability p is extremely low is due to the fact that we used a repeated measures t-test (paired t-test). This test considers the difference for the same subjects. If we follow the development for each of the participants (see appendix A) we can see that for almost all participants there is an increase in the performance.

However, we overestimated the final performance at the end of phase 1. The mean and its confidence interval are below the expected 80% for the last 12 comparisons. One of the reasons is that there is a strong variation between the performances of individuals at the end. Some participants have over 90% coherence in the last 12 comparisons. However, there are other participants who have less than 50% of coherent judgements. Since we did not cluster participants who performed badly or well we have a high variation. In order to cluster participants we need an independent variable that defines the clusters. For the evaluation of phase 3 we will develop a rating for the classification. This rating can be used as an independent variable and the coherence performance will be interpreted as the dependent variable.

4.3.5 Phase 3: Classification into full partitions[30]

In phase three participants were asked to create classes that do not overlap. For example, the patterns P1, P2 and P3 are a complete partition of G.

4.3.5.1 Research question and hypotheses

Since participants had to decrease the number of classes they can define, we can assume that they concentrate on the classes they are most confident in. The question is whether other people found the same patterns. That is whether they saw dominant similarities in the same sets of graphics.

H3.1: We expect that the patterns P1, P2 and P3 are found most frequently.
H3.2: We expect that the complete partition into P1, P2 and P3 is found most frequently.

Since we proposed P1, P2 and P3 as the patterns that fit best to the similarities and differences of the graphics, we also assume that the exact partition of P1, P2 and P3 is found most often as H3.2 states. Note that a participant who has found P1 and P2 could fail to find P3. For example, the participant could further subdivide P3. The number of possible complete partitions for a set of 14 items is the Bell number of 14: $Bell_{14}$=190899322.

4.3.5.2 Results

In total, 309 sets have been marked (an average of 4,41 sets per participant)[31]. These 309 sets consist of only 80 different sets, meaning that some sets have been frequently used. Table 9 shows the distribution of pattern sets. There is one set P_{x1} that has been found by 49 participants (=70,0%). P_{x1} = P1-1 is a subset of P1 and contains the rollover graphics. Also, there is a long tail of 48 sets that have been found only by one subject each, and 11 sets that have been found only by two subjects each.

[30] Some of the results of phase 3 have been published as “Lessons learnt in mining and writing design patterns for educational interactive graphics” in Computers in Human Behaviour, 25(5). The article only covered the classification of the first 3 groups (Kohls & Uttrecht, 2009).

[31] One participant sheet was not properly filled out in phase 2. To be consistent, this sheet was also excluded in the evaluation of this phase. Therefore the analysis is based on 70 participants.

Number of sets	Found by n participants	% partipants who found the set	Set index	Patterns
1	49	70,00%	P_{x1}=P1-1	Switching States (Rollover)
1	45	64,29%	P_{x2}=P1-2	Switching States (Clicks)
1	25	35,71%	P_{x3}=P2	Activator
1	18	25,71%	P_{x4}=P3	Active Areas
1	14	20,00%	P_{x5}=P2 + P3	Union of Activator and Active Areas
1	13	18,57%	P_{x6}	
1	10	14,29%	P_{x7}=P3-1	Active Area (images only)
1	9	12,86%	P_{x8}	
2	7	10,00%	P_{x9}, P_{x10}=P3-2	Active Area (text only)
1	6	8,57%	P_{x11}	
2	5	7,14%	$P_{x12..x13}$	
2	4	5,71%	$P_{x14..x15}$	
6	3	4,29%	$P_{x16..x21}$	
11	2	2,86%	$P_{x22..x32}$	
48	1	1,43%	$P_{x33..x80}$	

Table 9. Distribution of identified sets (complete partition)

Obviously, there are some pattern sets that are dominating and have been chosen frequently. ACTIVATOR was found by 25 participants (= 35.71%), ACTIVE AREAS was found by 18 participants (= 25.71 %). Some people divided ACTIVE AREAS into two subsets, depending on whether an image (10 participants = 14,29%) or text (7 participants = 10,00%) had changed in the graphic example. Another 14 participants (=20,00%) saw ACTIVE AREAS and ACTIVATORS as instances of the same set – which makes sense because these two pattern sets do not discriminate over all interaction attributes. The most interesting results, however, were again found for SWITCHING STATES. Not a single person marked a set that contained all graphics that we had considered as examples for this pattern. The more specific patterns had been marked 49 times (= 70,00 %) and 45 times (=64,29%) respectively.

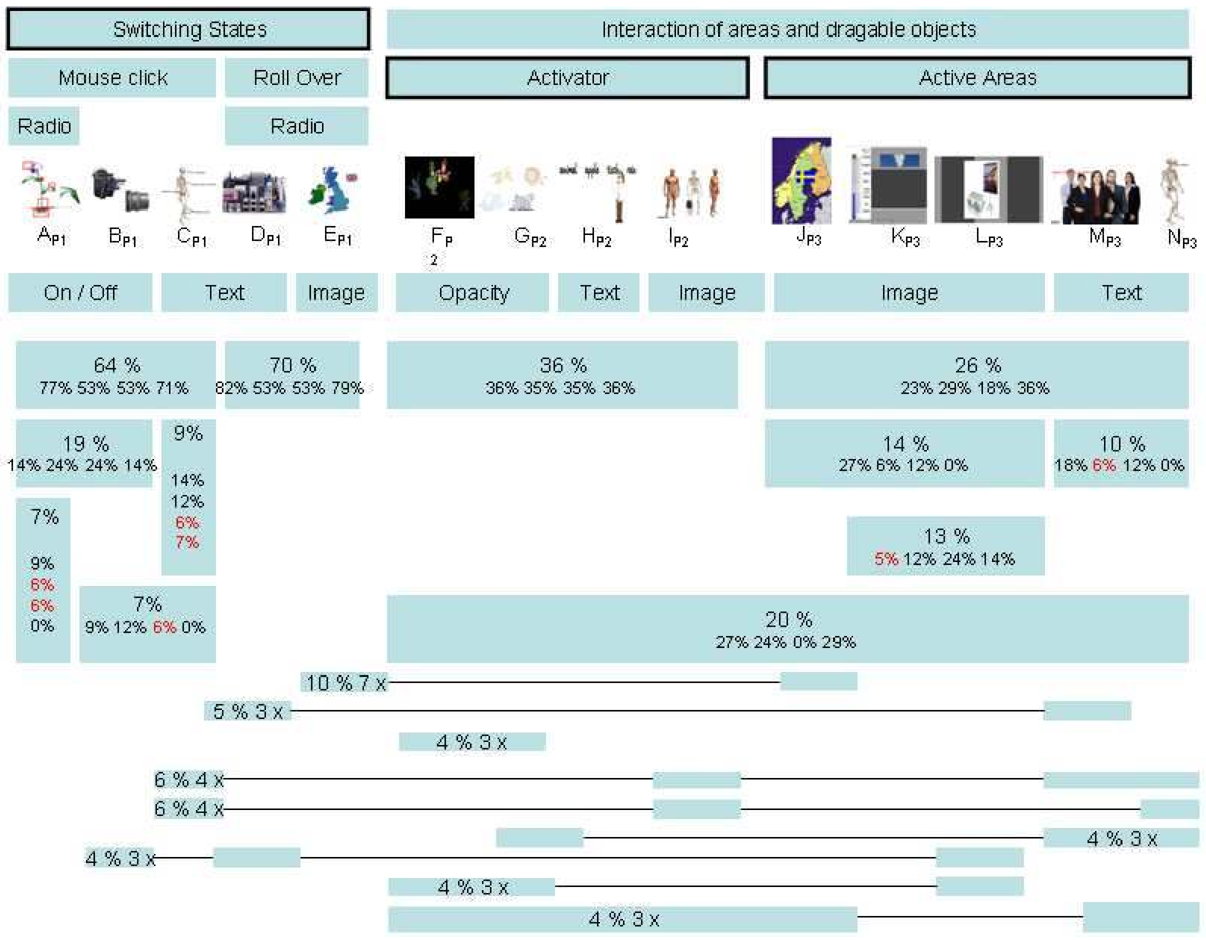


Figure 85. Identified sets (complete partition)

The complete partition into P1-1, P1-2, P2, and P3 was identified by 17 participants (24,3%). Another six participants found a similar partition but sub-divided P3 further. Four participants identified the partition into P1-1, P1-2, P2, P3-1 = {J_{P3}, K_{P3}, L_{P3}} and P3-2 = {M_{P3}, N_{P3}}. Two participants identified the partition into P1-1, P1-2, P2, P3-3 = {J_{P3}, M_{P3}, N_{P3}} and P3-4= {K_{P3}, L_{P3}}.

The complete partition P1-1, P1-2, and P2+3 (Activator and Active Area as one class) was identified by 10 participants (14,29%). Another three participants identified a similar partition but considered the graphic A_{P1} as a separate class.

There have been further 35 partitions that only have been identified by a single participant each. Note that any recurrent complete partition is very unlikely by chance because the set of 14 graphics allows 190899322 different complete partitions (see appendix C).

4.3.5.3 Discussion

In general, the patterns we had mined before were the most dominant. As mentioned before, individuals are likely to create their own patterns in mind. But in documenting a pattern, one wants to make sure to find a pattern that is understood and agreed upon by many people. The result showed us that we had to document each variation of the STATE SWITCHER pattern separately. This was an important finding, because it was for exactly this pattern that we had been unsure about which level of abstraction to choose. Whether an interaction is triggered by a mouse click or a roll over shows to be a discriminating criterion. As a result, the pattern subset P1.1={D_{P1} , E_{P1}} contains only two elements. These two elements, however, are distinct regarding the number of property values for the output element. So maybe, if we had chosen more example graphics that are distinct on that dimension, the pattern subset would have been further partitioned. In the same way, in P1.2={A_{P1}, B_{P1}, C_{P1} } the element A_{P1} is distinct to the other elements in that sense that only one of the output elements can be activated at the same time. Would P1.2 have been split up into further subsets if there was another graphic with this feature? What we have learned is that all dimension of interaction are strong discriminators. So, does it make sense to split up the STATE SWITCHER pattern into sub patterns RADIO BUTTON = { A_{P1} }, ON/OFF BUTTON = { B_{P1}, C_{P1} }, ROLLOVER BUTTON = { D_{P1} } and INFORMATION DISPLAY = {E_{P1} }? To test this hypothesis, one has to design another study in which the example graphics include more than single instances of the patterns. The interaction dimensions are not equally strong. If that would be the case, there should be no set that consists of elements from both ACTIVATOR and ACTIVE AREAS because the elements differ on three dimensions of interaction. But 20% of the participants have ignored these differences and put the graphics into the same set. Indeed, more people have chosen to put them into different sets and used the same discriminating dimensions for their judgements.

It is important to note that a pattern set P_x found by the participants is only based on the example graphics given in the study. When we proposed that P2 = {F_{P2}, G_{P2}, H_{P2}, I_{P2} } contains instances of ACTIVATORS, we had the pattern in mind and were familiar with many exemplars of the pattern. A participant who has found the set P2, however, was only familiar with the example graphics of G. It would be interesting to see what happens if other instances were taken as material and how the number of instances for each pattern influences the result.
Also, we have to design another study in which different patterns are tested. A more serious issue is that the example graphics in G only consist of one pattern each. Many interactive graphics in the real world use multiple patterns at the same time. The patterns only assign roles to the graphic elements. Each element can act in multiple roles at the same time. This would make it harder to identify and describe the patterns.

The study showed that people tend to group interactive graphics into some dominant sets. But it also showed that people use different criteria to distinguish between two types. This classification, however, is limited to the given material and it only captures similarities in the form. The problem statements and contexts of application were not treated explicitly.

4.3.5.4 Cluster analysis of classification

If we look at the classifications given by the participants we will find some classes that are very similar to each other. For example, consider the two classes PA={J_{P3}, L_{P3}, M_{P3}, N_{P3} } and PB={J_{P3}, K_{P3}, L_{P3}, M_{P3}, N_{P3} } which were found by two different participants. Since PA and PB are not exactly the same they are not counted as recurrent found classifications. However, PA and PA are obviously of a similar kind – there is a class of similar classes.

The grouping of similar but not identical objects is the subject of cluster analysis. A cluster analysis divides a number of objects in a hierarchy of classes and sub-classes. Within a class the objects should be homogeneous; the objects of two distinct classes should be different. This is similar to our assertion that graphics within a class should be homogenous. In this analysis step, however, we are no longer comparing the similarity of interactive graphics but the similarity of the classes.

We can define the distance between two classes by the number of members that are found in one class but not the other. The maximum similarity is given if two classes define exactly the same set of interactive graphics – the distance between the two classes is 0. The maximum distance is given if two classes define sets that contain all interactive graphics but no interactive graphic is part of both sets. For example, the distance between P0 = { } and P14={A, B, C, D, E, F, G, H, I, J, K, L, M, N } is 14; likewise the distance between P9={A, B, C, D, E, F,G,H,I } and P5={J, K, L,M,N}, is 14 because none of the seven member of P9 is found in P5 and none of the five members of P5 is found in P9.

A hierarchy of clusters is built either by splitting up one cluster that contains all objects into sub-clusters which are then recursively split to sub-clusters down to the level where each cluster only contains one object. More common, however, is the bottom-up approach where each object is first its own cluster and pairs of clusters are merged to higher-level clusters.

There are different strategies to merge two clusters into a higher-level cluster. Each strategy investigates the distances between the different clusters. The strategies differ in the way the distance is calculated:

Average Linkage: The distance between two clusters is the medium distance between all the objects of the two clusters.
Single Linkage: The distance between two clusters is the smallest distance between objects of the two clusters (nearest neighbour)
Complete Linkage: The distance between two clusters is the largest distance between objects of the two clusters (furthest neighbour)
Ward Method: The criterion to merge two clusters is to minimize the variance of objects in the higher level cluster.

The dendrogramms in figure 86 are showing the hierarchical order of clusters calculated based on the different strategies. The calculation and generation of dendrogramms was done with SPSS. The first dendrogramm shows the hierarchy of the expert classification (see figure 78) as reference. Circles indicate the different classes based on interactivity. The boxes at the bottom show an alternative classification based on superficial features such as the property that is changed.

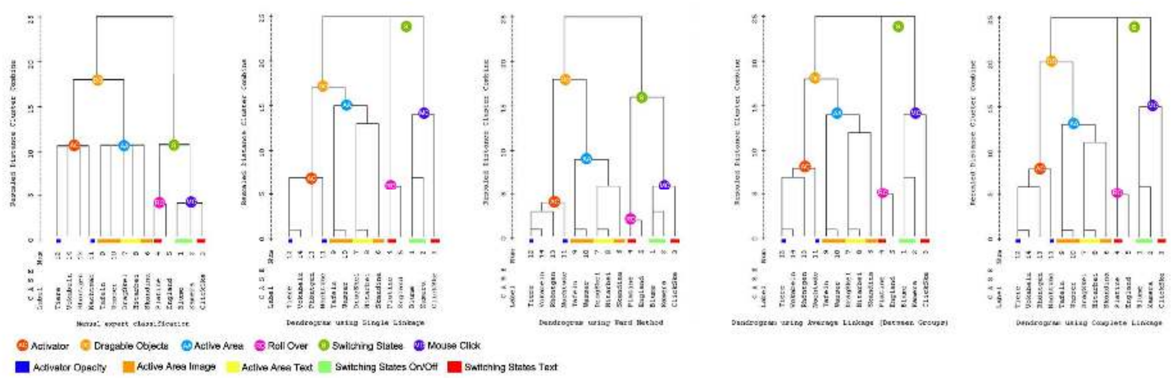


Figure 86. Hierarchy of classes based on cluster analysis

The dendrogramms almost all show the same hierarchy of classes but different distances between the branches. This must be so because the distance is defined in different ways. The most important fact is that the structure reflects discriminatory features between classes on different levels in the hierarchy. Technically, the major difference between the calculated dendrogramms (using cluster analysis in SPSS) and the expert classification is that the expert's classification does not calculate any distances (the distance is only estimated) and a branch can split into any number of sub branches (e.g. Activator splits into four graphics) whereas the calculated clusters always splits one cluster into two (top-down) or join two clusters (bottom-up), sometimes at the same level. Therefore, the calculated dendrogramms also show clusters with only two graphics as members at the bottom whereas the expert classification does not subdivide the classes to this fine level. In general there are more levels of branches (a class is more often divided) because the relevance of discriminatory features is not accounted. It is less obvious which splits are more important. For example, all dendrogramms have a branch that connects sub-branches directly to graphics of the ACTIVATOR pattern. The calculated dendrogramms, however, further subdivide the clusters according to superficial features (e.g. which property – opacity, text, image - changes).

Since the dendrogramms are based on the classes identified by the participants they inherit such differentiations. Yet it is important to see that participants made these specific differentiations because sometimes they point to relevant features that have not been accounted for in the expert's classification. For example, the set of "Switching States" graphics is further divided into "Roll Over" and "Mouse Click" in all cases (including the expert's classification). Yet only the calculated dendrogramms show that "Mouse Click" is consistently subdivided into "ON/OFF" and "Text". That means that many participants differentiate between switching an image or a text label on/off.

4.3.6 Performance difference for canonical and non-canonical classification

Phase three has shown that some classes are found more frequent than others and that there are also some complete partitions that recur. We can consider the partition that occurs most frequently as canonical, i.e. agreed on by many persons.

Since overlapping classes are forbidden in this phase of the experiment, participants were forced to make a decision to which class a graphic should belong. Hence, these classes represent the schemas a participant is most confident in. If we assume that the reason that some partitions are found more frequently is based on a higher homogeneity within these classes, then we should expect that persons who found the canonical classes have build schemas that are more coherent. Hence, for those persons we can expect that their pair comparisons are more coherent towards their individual non-overlapping classes.

4.3.6.1 Research questions

Because we assume that there actually is a higher degree of similarity within the canonical classes, for a person who has induced this classification it is more likely that her or his classification is based on similarities that have been recognized during the pair comparisons. A person, however, who had found a different classification, had most confidence in schemas lacking this homogeneity between exemplars. For these persons we should expect that their similarity judgments in the pair comparisons are in conflict with their non-overlapping classification. If this was not the case there would be indeed alternative homogeneous classifications of our graphics, i.e. we could coherently classify the given set of graphics based on other discriminatory features.

Levels of classification quality

To test these assumptions we will consider the performance for participants that have found

- the canonical classification (A rating),
- a classification close to the canonical (B rating),
- a classification that is different in some aspects to the canonical (C rating), or
- a classification that is very different to the canonical (D rating).

Classification, here, means the complete partition into classes. That is, the canonical classification consists exactly of P1-1, P1-2, P2, and P3. To define which type of classification each individual has found, we can count the number of deviations or distances to the canonical classes:

- If all classes of the participant are the same as the canonical, the distance is 0.
- If there is a class similar to a canonical class, then for each graphic missing or each graphic not belonging to the canonical class, we increment the distance by one. That means if one graphic is put into another canonical class it will cause a distance of two because it is missing one time and falsely included another time. If a graphic is put into a new class that is not found in the canonical classification it only causes a distance of one.
- A split up of a canonical class into two sub-classes is counted as one deviation, causing a distance of one. The graphics that are correctly found in the sub-classes of a canonical class are not counted again as a deviation to the canonical super-class. However, if a participant's sub-class is only similar to a sub-class of a canonical class, then each difference counts as one deviation.
- Likewise, a merge of two canonical classes to a super-class is only counted as one deviation. If graphics are missing or added to the super-class, each difference again increases the distance by one.

A-classifications are exactly like the canonical classification and have 0 deviations. B-classifications have 1-3 deviations. C classifications have 4-6 deviations. D classifications have 7 or more deviations. For example, if somebody has put just a single graphic from ACTIVATOR into ROLLOVER, this would cause a distance of two (because it is missing in ACTIVATOR and falsely in ROLLOVER). If somebody merges ACTIVATOR and ACTIVE AREAS into one class it only causes one distance (the classification is very close to the canonical). Even if one of the graphics of this non-canonical super-class is put into ROLLOVER, it would still be a B-classification (distance of three). However, if more differences occur, we can no longer say that the classification is close to be canonical.

Expected correlation of classification rating and judgment performance

We expect that participants who have found the A rated classification will perform better than participants who found B rated classifications, and those who have found B rated classification perform better than those who have found C rated classifications, etc. Somebody who has found the A rated classification should have a higher percentage of similarity judgements that are coherent with that classification.

As for phase 2 we can define a function that returns the mean of coherent judgments. The coherence can be measured in the same way as for the overlapping classes in phase 2. This time, however, we check the coherence towards the non-overlapping classes defined in phase 3. Since we are interested in the different means for participants who found A, B, C, D rated classifications we need a third parameter for the function coherence (a, b, c), where a and b define the range of pair comparisons considered and c defines which participants are considered.

4.3.6.2 Hypothesis about the different performances

We expect that the overall performance is better for those participants closer to the canonical classification:

H3.3: coherence(1,103, A rated classification) >
coherence(1, 103, B rated classification) >
coherence (1, 103, C rated classification) >
coherence(1,103, D rated classification)

To check the statistical significance we can use the unequal variance t-test. Since we do not know the variance for each of the samples and we cannot assume that the variance for each group is homogeneous, we need to be conservative and use the unequal variance t-test. Since we are confident that the canonical classification is due to a higher coherence between the class members, we expect that this coherence is reflected between the pair comparisons and the classification. Hence, we only accept the hypotheses if the results are very significant (alpha < 0,01).

4.3.6.3 Results

The outcomes of H3.3. are shown in table 10.

	A-rated Classification	**B-rated Classification**	**C-rated Classification**	**D-rated Classification**
Mean	86,58%	83,09%	68,64%	61,41%
Standard Error	0,12	0,14	0,22	0,24
P(T<=t) one-tail		0,000450531**	1,69246E-19**	0,000600608**

Table 10. Correlation between performance and classification quality

The assumption that the quality of classification has a positive impact on the coherence of judgements (H3.4) holds in our experiment. Note that we included all pair comparisons in the t-test rather than the means of each participant. The low p value is due to the high sample size that we get if all pair comparisons are included (see appendix A).

4.3.6.4 Discussion

We argued that many different classifications or partitions are valid because people can build different schemas. However, we expected that some classifications (or schemas) are more coherent in respect to interaction features and their structural relations.

If there are classifications that are more reasonable than others, then those classifications should be found more frequently. Therefore, we considered the canonical classifications as more reasonable. This classification seems indeed to be more reasonable because people who found it made more coherent similarity judgements in the phase of pair comparison. That means the classification is actually based on the identified similarities in the first phase.

The canonical classes are characterized by members that maximize coherence between each other. We can therefore argue that these classes are the most interesting ones to be described. That does not mean that other classifications are wrong. However, other levels of abstraction make it hard to find coherent descriptions. If we choose to describe a more abstract pattern we need to include many variations. A BUTTON pattern needs to include the diverse button types that exist. An ACTIVE AREA pattern still needs to point out which variations exist (e.g. images or text can change). However, these minor differences do not justify the distinct pattern descriptions. While BUTTON is too abstract, ACTIVE AREAS is just right. The category of BUTTONS includes too many different gestalts to coherently describe the universal structure. The interaction structure OF ACTIVE AREA, however, does not differentiate further except for the type of property changes (see section 4.4 for a discussion of the evolution of the pattern descriptions).

4.3.7 Phase 4: Naming of classes

The fourth phase of the experiment was very simple: participants had to find expressive names for each of the non-overlapping classes found in phase three. The names were added on the sheet where the classes had been encircled.

4.3.7.1 Analysis of the outcome

The next tables show the names for the patterns that have been found most frequently. The table includes all labels. If the same label has been used by different participants it appears multiple times. The names have also been grouped by similarity. The comment highlights what the name implies and which interactive features are covered by the name.

Names for the pattern ON / OFF BUTTON	Comments on the name's implication
Click! / Click / Click / Click / Click / Click / Click Click class / Clicking / Click me / Click at / It clicked. / CWD – Click without drag / Click and stay / Click makes smart / Click yourself smart / Point and Click / Click & Change or Click and variation	Click event
Click and show / Show-Click/On-Off / Show and Hide by Click / Click – Show / Click -> Fade in	Click shows something
Infoclick / Click-to-info / Clicking new image (info click)	Click shows information
Click and find out / Click to find out more / Click explosion	Clicking at areas or positions causes an action
Explanation by clicking at area / Action by click at static area / Background changes by clicking at certain positions	Click provides details
Details by Click / Show details by click / Zoom and Explain Click / Zoom / Zoom / In Detail / Detail view	Details are provided
Reveal animation / Clear to the point / Labeling features / Component cursor	Name implies what changes
Selectable elements (static) / Multi-selection / Stay there	Name implies something can be selected

Names for the pattern ROLLOVER	Comments on the name's implication
The mouse knows it / Mouse visible / Mouse selection The smart cursor graphic / Cursor on Background changes by positioning the mouse over certain areas	Mouse action
Move the mouse and find out more , Explanation by rollover of mouse pointer / Move mouse with info / Cursor slide provides info /Mouse dragging gives info	Motion of mouse
Mouseover / Move over / Mouse-Over / Mouse over / Roll-Over / Mouse over effect	Mouse over
Explanation tooltip , Mouse over / Mouse over image (Mouse Over) / "Info" by "mouse over"	Mouse over and effect
Hover / Hover / Explanation by hover	Hover
Move over me / Move over for infos	Motion over something
Free-move-to-info / Come there and find out / Go	Motion/Positioning to get information
Detail by point at / Electronic pointer / Simple-Point / Pointing -> fade in / Arrow move class	Point at something
Search class / Searcher / Selectable elements (volatile) / As you like it	Selecting/Searching
Without click / Single answer without click / Drag mouse only over image / Just change or Change without request or Variation without command / MW – Mouse without	Changes or effects without explicit click action

Pop-On-Animation / Labels / Show and name / Display details by touching	Display information

Names for the pattern ACTIVE AREA	Comments on the name's implication
Changes of pointer / Icon change info Adaption to topic / Variety Cursor or Variable Pointer (to touch) / Transitions	Changes of an object or pointer
Pulling causes change / Cursor pull changer / „Click and pull" -> "target object" changes / Change of an object by moving it / Move graphic and source changes (source change) / Drag me and I change for you / Drag symbol and change class	Pull/Move/Drag causes change
Draggable objects changes by positioning/ Object changes when moved over background / On a static figure something is moved and changed	Pull/Move/Drag causes change based on background position
Object over area provides info for the background in question / Changeable pointer, constant background	An object changes based on background
Foreground / Mouse pointer / CWM – Click with pull	Name does not fully imply what happens

Names for the pattern ACTIVATOR	Comments on the name's implication
Changes in the background / Background change info / Variety Background or Same Pointer – Variable Background or Pairs (one reveals images)	Background changes
Move object highlights background / Background changes by moving an object / Background changes if object moves. / Drag Cursor Background Changer	Background is changed by an object/cursor/pointer
Cursor changes graphic / Change of an image by dragging a figure / Drag symbol and target change class / Move image -> target changes (target change) / Click and drag -> graphic changes	Graphic is changed by an object/cursor/pointer
Constant Cursor, changeable background / Background-Change (Moving object does not change) / CWD – (Click with drag) Background	Background is changed, moving object is constant
I change the graphic for you / Image, change!	Changes of an image/graphic
Take + Move = Information / Dragging brings knowledge / Exploring class Become visible / Move-Explain	Interaction reveals information
Background / Drag only by keeping / Cursor image for component / Move Graphic	Name does not fully imply what happens

Names for the pattern ACTIVE AREA (image variant)	Comments on the name's implication
Change class / Cursor-Morpher	Object changes
Change by motion / Drag graphic that changes Foreground change (drag element changes)	Object changes by motion
Drag image / Image drag technique / Click and drag image	Image is dragged
Visualization / Optical illustration	Visual impact

Names for the pattern ACTIVE AREA (text variant)	Comments on the name's implication
Label class	Label
Drag and label | Drag with change of labelling | Click & Drag Text	Text/Label can be dragged
On dragging to an object -> Explanation | Drag controller changes animation Infocursor	Dragging an object

Table 11. Pattern names expressed by participants.
The names have been translated from German to English.

4.3.7.2 Discussion

The similarity of the names given to the classes show that the participant understood what happens in the graphics and what discriminates them from each other. The names imply that the classification is based on interaction principles. Some participants even used technical terms such as Hover or Mouse Over[32].

Most of the names focus on the interaction. However, some names focus on the effect the graphic has (such as zooming, provide information, labelling, or exploring) without telling us what actually happens.

The names also explain some of the non-canonical classes that have been found. For example, a class that includes the graphics that show Great Britain (E_{P1}) and the Scandinavian countries (J_{P3}) has been found seven times (10%). We would expect that the graphics have been grouped because they have the same topic. The names support such an assumption. Participants labelled the class: Countries, Knowledge about flags, Countries / Flags, Geography, and Maps.

Likewise, a class that showed technology products (camera, motherboard, black boards) was found three times. Each time it was labelled "Technology". Another class was found four times: $P=\{C_{P1}, N_{P3}, M_{P3}, I_{P2}\}$ contained only graphics that showed human beings. Hence, its labelling: Human, The Human, Humans, Human. M_{P3} of that class shows employee while all other graphics show medical images such as X-Rays of a person or a skeleton. This explains why a sub class $P=\{C_{P1}, N_{P3}, I_{P2}\}$ was found that had been labelled Medicine, Human, or Anatomy.

Three participants had found $P=\{F_{P2}, G_{P2}, H_{P2}\}$. This seems to be an arbitrary part of P2 – the ACTIVATOR pattern. The names, however, suggest that the drawing style had influenced the classification. The class has been termed Comics, Kids stuff, and Explanations. In general, the non-canonical classes are often not based on interaction features. The graphics have been grouped by subject, image type, style, or teaching goal.

By analyzing the names we can see that some of the participants had been aware of the hierarchy of classes. For example, one participant has identified the sub classes of ACTIVE AREA. We can see that the participant is aware of the connection by looking at the symmetry of names:

> *"Moving picture that changes (water, boards, scandinavia)"* [ACTIVE AREAS, image variant]
> *"Moving with text changes (drag skeleton, employee)"* [ACTIVE AREAS, text variant]

Another participant has identified both ACTIVATOR and ACTIVE AREAS and acknowledged their higher order relation in the names:

> *"Background changes (moving object does not change)"* [ACTIVATOR]
> *"Foreground changes (moving object changes)"* [ACTIVE AREAS]

4.3.8 Phase five: Description of a class

The last phase of the experiment asked participants to pick up the class they were most confident about and describe it. As a guide the participants were asked the following questions: How does the interaction work? Which elements participate? Which functions do the elements have? How is the interaction different to other classes? What other applications are there for this interaction? What are examples you could think of?

4.3.8.1 Research questions

Once again we can assume that participants who had identified more coherent and homogenous classes (such as the canonical classes) will perform better than others because the descriptions can refer to generalized principles.

To test this assumption, each description was rated for the following dimensions that reflect the guiding questions:

- Is the functionality of the interaction fully described?
- Is every participating element mentioned?
- Are the roles and functions for each element described?
- Is the interaction compared to other classes?
- How many examples are given?
- What is the overall quality of the description? Does the description use clear and precise language?
- Does the description abstract from the given examples of interactive graphics?

[32] The German participants used the English terms "Hover" and "Mouse Over".

The last question checks whether participants generalize over the functionality and elements (e.g. "there is an element to click on") or refer to the specific instances (e.g. "there is a camera one can click on"). If the interaction is described in general terms it is more likely that the participant has induced a schema rather than referring to episodes.

The ranking was mapped to numerical scores (1 = very good; 6 = insufficient) for each of the dimensions. The scores of the single dimensions were mapped to a total score between 1 and 6.

4.3.8.2 Hypothesis about the relation between classification level and description quality

We assume that the mean of the total scores is better for participants who had found canonical classification. Let score(c) be the mean of all scores from participants who had found classifications of level c. Then,

H5:
score (A rated classification) < score (B rated classification) <
score (C rated classification) < score (D rated classification)

4.3.8.3 Results

Before we consider the different scores for the classification level we will consider some example descriptions.

This is an example of a description of a participant who found the canonical classes and described the Activator pattern:

Name: Change of target (Move picture – target changes)
Description:
a) one clicks at an object and moves it via drag and drop
- when the target area is reached, the graphic at that location will change (graphic will change if a target graphic is reached)
- there can be multiple target graphics, each target graphic does have a second graphic
b) Pointer graphic
Target graphic (1-n)
Replace graphic (1-n)
c) Pointer graphic has no interaction function, only provides a matching picture
Target graphic (1-n) waits to be touched by pointer graphic
Replace graphic (1-n) will be shown instead of target graphic (1-n)
d) Drag and Drop (picture will be replaced)
The target will be replaced
e) This interaction could replace all other interactions, at least its applicability is most flexible

The description is very brief yet all questions are answered precisely. All roles have been listed and described. Moreover, the number of possible elements that could play that role is given. It is noted that there needs to be an alternative graphic to each of the target graphics. The comparison to other interaction classes highlights the most important differences: while the click and rollover buttons depend on the mouse pointer this interaction depends on drag and drop. It is also different to ACTIVE AREAS in that the target will be replaced rather than the pointer graphic.

14 of the 71 participants have chosen to describe the pattern P2-3 which is the super class of ACTIVATOR and ACTIVE AREAS. The following description is from a participant who gave an average description and did not find the canonical classification:

Name: Changing drag image
Description: This class consists of an image that can be dragged (=drag image). For this "drag image" there are multiple images/areas/words. If the "drag image" is dragged to another image/area/word there will be a change at that position.
For two images of this class the drag image changes if dragged onto the area. ***Actually one could say that is a sub-class.***

In phase 3 the participant had chosen the higher level pattern. The description acknowledges that there are specializations of the pattern ("Actually one could say that is a sub-class."). This shows that some participants were aware of more relations between the classes but did not clearly express it in their classification. Unfortunately the actual differences are also not clearly expressed in the description which is one reason for being rated avarage. While the description includes all important graphical elements it is unprecise about the actual nature of the change. It remains unclear whether the element in the background always changes or only if the drag element does not change. The number of elements that can change (1 or n) is not specified and the the number of states required for changeable elements (2 or n) is not specified. The description also refers to specific examples of the test instead of describing the behaviour in more general terms (e.g. refer to two exceptions of the interaction form, and listing property examples instead of generalizing to any change of a graphic).

Finally, we can see that participants who had made classifications very different to the canonical had trouble to describe the general interaction of one of their patterns. For example the next description covers a class that has been found several times but does categorize the graphics by subject rather than interaction:

Name: Technology
Description:

a) all members of this class are technical objects
b) PC, Projector, Camera
c) Camera: Record images
Projector: Show images
PC: Store, process and show images/data
d) Clear display of technical components, information about labels of single parts and respectively the development of a technology
e) *Displaying historical events along a "timeline"; explanation of technical components, stand-alone information systems*

We can see that the interaction is not described at all. Moreover, the description does not abstract but refers to the particular examples without identifying structural invariants.

The ranking and correlation to the classification quality is given in table 12. The classification quality is based on deviation to the canonical classes (see section about phase 3).

	A-rated Classification	**B-rated Classification**	**C-rated Classification**	**D-rated Classification**
Mean	1,98	2,20	2,87	4,02
Standard Error	1,32	0,74	1,39	0,45
P(T<=t) one-tail		0,25173718	0,06098298	0,00794936 **

Table 12. Correlation between quality of description and classification quality.

The table shows that the score differences between A and B rated classifications and B and C rated classifications are not significant. Only the score difference between C and D is significant. The difference between A and C (1,98 vs. 2,87), too, is significant (p=0,0358467).

4.3.8.4 Discussion

The quality of descriptions correlates with the quality of classification. However, the effect between two classification levels is too small to be significant and not by chance.

The canonical classes are not always better described. Only very poor classifications (D-rated classification) significantly correlate with weak descriptions. For this we can find two reasons. A non-canonical classification makes it harder to find out similarities of interaction. Descriptions of non-canonical classes refer more frequently to particular objects instead of using generalized terms. Moreover, since the functionality of the class members is different, the description of the functions often misses many details.

The other reason for descriptions of lower quality is that participants may not have built appropriate schemas at all. They might need further examples to induce stable schemas. If one does not have a schema one can hardly describe it: Explication of implicit structure is hard; explication of implicit structure not built is impossible.

However, there are also some very good descriptions of participants who have found non-canonical classifications. One of the reasons is that participants were allowed to pick up the class they wanted to describe.

Hence, a participant who made a non-canonical classification but found at least one canonical class might have picked it for the description. S/he would be expected to perform well for that class. It is notable that most participants picked a canonical class – even those who had not a canonical classification. This could indicate that some classes were easier to be found or clearer distinguishable from other classification options. There is also the danger that participants picked up the class that required the minimal effort to describe.

4.3.9 Conclusions

We designed the experiment in order to test whether the patterns P1, P2 and P3 were defined at the right level of abstraction and granularity. In each of the first three phases it emerged that the abstraction level of P1 has to be rejected. If we had proposed two more specific patterns P1-1 and P1-2 instead, we would have found our hypotheses confirmed. This has implications for the description of patterns. It is more effective to describe P1-1 and P1-2 as distinct patterns. The outcome of this experiment was the reason we re-wrote the pattern STATE SWITCHES and split it up into ON/OFF BUTTON, ROLLOVER BUTTON, RADIO BUTTON and INFORMATION DISPLAY. Those patterns are similar technically but have different effects, consequences and contexts of applications as we will see in the next section. While the patterns overlap partly, each interactive graphic could be characterized best as an implementation of one of these (A_{P1} is a RADIO BUTTON, B_{P1} and C_{P1} are an ON/OFF BUTTONS, D_{P1} and E_{P1} are ROLLOVER BUTTONS).

4.3.9.1 The way participants learned

Phase 1 and 2 show that the schemas the patterns are based on are actually learned over time. This is evident in the increasing performance of coherent pair comparisons.

Moreover, it turned out that our participants learned similar things from the given graphics. As expected, there is a level of abstraction that is shared by most people. P1-1, P1-2, P2 and P3 are the most frequently found classifications. The classification has been done independently by each participant and still the same classifications emerged. This implies that these schemas are meaningful to most of the participants.

4.3.9.2 Different schemas emerged

Yet there are some differences in the schemas induced. For example, some participants preferred an abstraction of P2 and P3 into P2+3. It also showed that only the sub classes of our proposed pattern P1 were meaningful to the participants.

There are alternatives to the canonical classification. Some schemas occurred often enough to be significant. The naming of phase 4 explains the background of these classes. They are usually not based on interaction but subject, style or effects. They are not wrong but less appropriate to capture homogenous classes of interaction. This interpretation is supported by comparing phase 1 and 3. The performance of pair comparisons is more coherent if canonical classes were found. This implies that the members of the canonical classes are indeed more homogeneous - at least in the dimensions of interaction. As a result the general interaction is harder to describe for non-canonical classes as phase 5 shows.

4.3.9.3 Quality of classification

The categorization into other classes is not wrong but less appropriate to capture the actual structure of an interaction. For example, a class that categorizes by subject does not tell anything about the actual configuration. We can say that these classes rather consider the similarity of the macro-state instead of the micro-states. Hence, they are less informative because they are telling us less about the actual structure. To say we need a ROLLOVER BUTTON is more instructive as to say we need a "Show details" graphic. The reason is that "Show details" can have many different gestalts whereas the range of possible forms of a ROLLOVER BUTTON is constrained. This is similar to our example in section 2.4.2.3 where the instruction "Cross the river" is less informative than a description of a specific path. There are many ways to achieve a specific pedagogical goal, but listing all the ways one by one is not helpful.

4.3.9.4 Classification based on second order features

To summarize the outcomes, we have seen that graphics have often been judged by deeper structure instead of superficial features. Judgements became more coherent over time as the schemas finally became relatively stable. That different schemas were found shows that the classification task was not trivial and that multiple perspectives to the given graphics are possible. There are, however, canonical schemas that more people agree on. These schemas are more homogeneous and they are more straightforward to describe.

The next section will report about the experience of describing patterns and how the descriptions and pattern formats evolved.

4.4 The description of patterns

The previous sections have discussed the way of finding, implementing, and experimentally testing the patterns of interactive graphics. This section will discuss the explication of patterns in written pattern descriptions. We will trace how the description format that we used for patterns of interactive graphics has changed over the last years of publications on PLoP and EuroPLoP conferences. Since some of the patterns have been rewritten or split up into various specialized versions, we can discuss the changes of the description format, the changes between description versions of the same pattern, and the changes if a pattern is discussed on a more or less abstract level.

Since we experimented with various description formats, we will use the standard descriptions discussed in section 2.3 as a reference:

- Name
- Context
- Problem
- Forces
- Solution
    - Details
    - Diagram
    - Examples
- Consequences

Since all pattern formats elaborate the solution in a discussion, exemplify the solution in examples, and often represent the core structure in a diagram, we will discuss the changes of these sub sections as well.

In our discussion of the structure of patterns we also elaborated on pattern languages. While there are many relations between patterns of interactive graphics, the collection is far from being a language. The patterns are not related in a way that one pattern builds on other patterns. Rather, the relations include specializations (e.g. TRANSPORT OBJECTS is a specialization of SYNCHRONIZE OBJECT MOTION, and there are several specializations of buttons), competing alternatives (different button types, different ways to restrict the motion area), and categories of purpose (motion, motion restriction, highlighting etc.). To illustrate the difference between single patterns and pattern languages we will consider a pattern language of online trainings and compare the different ways the patterns of a collection and the patterns of a language are described.

4.4.1 Evolving the pattern descriptions of interactive graphics

4.4.1.1 Different descriptions of the same patterns

All of the patterns that we described are based on the behaviours that are found in moowinx. While behaviours are technical implementations of interaction patterns, the described design patterns reason about the contexts of application, the pedagogical problems solved, and the forces that imply the specific form of the solution. In my master thesis I had described many example problems of implementing interactive graphics. The problems collected in that work did not address the reasons for a specific form of interaction; rather they described specific forms of interaction and analysed the involved roles of elements. They addressed the problem of implementation without reasoning why the interaction form was needed. This lack of information – the answers to why and when a specific form of interaction is useful – was a challenge for end-users of moowinx as we observed in trainings. This was a motivation to explain the problems that can be solved with the interaction forms and list appropriate applications in the written pattern descriptions. These patterns have been published at PLoP 2006 (Kohls & Windbrake, 2006), EuroPLoP 2007 (Kohls & Windbrake, 2008a), PLoP 2007 (Kohls & Windbrake, 2007), EuroPLoP 2008 (Kohls & Windbrake, 2008b) and EuroPLoP 2010 (Kohls, 2010a). Each collection of patterns has been shepherd in several rounds, run through a Writer's Workshop, and have been revised for a final version.

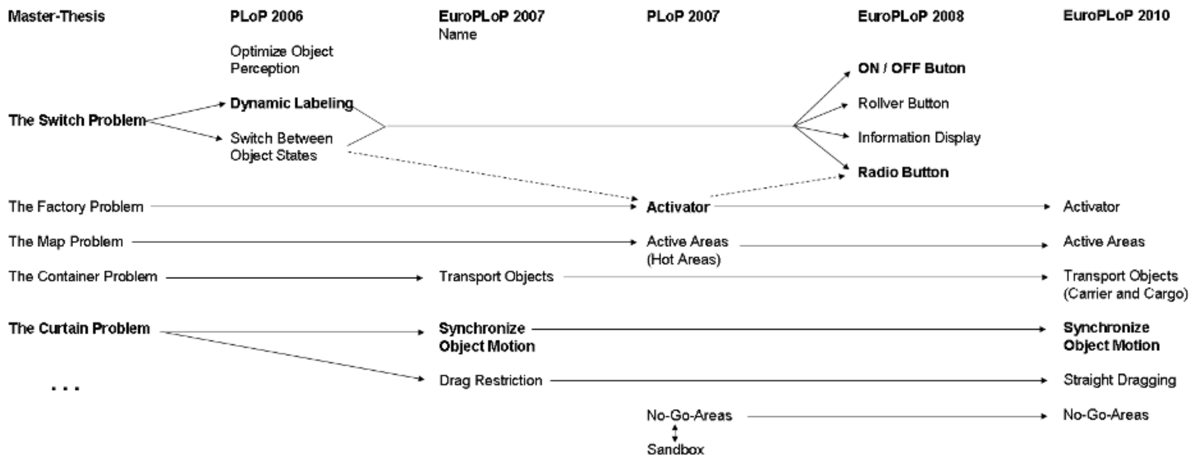


Figure 87. The patterns have been discussed on several PLoP. Some patterns have been published as different versions.

We will discuss the changes for pattern versions highlighted in bold in figure 87. Thus, we can follow how the descriptions for patterns of two different kinds (switching property states, and motion) have changed over time.

4.4.1.2 Development of different pattern formats

The use of different pattern formats is quite common. Different pattern authors make use of different pattern formats but one can always find a description of context, problem, forces, and the solution. Some examples are:

• "Patterns for Classroom Education" (Anthony, 1996) use:
P= {Name, Problem, Constraints and Forces, Solution, Related patterns}
• Some patterns from the Pedagogical Patterns Project (Bergin, 2001) use:
P = {Name, Thumbnail, Audience / Context, Forces, Solution, Discussion / Consequences / Implementation, Special resources, Related patterns, Example instances, Contraindications, References}
• The online repository of the E-LEN project uses:
P= {Name, Maturity level, Category, Problem, Analysis, Solution, Known Uses, Context, References, Related Patterns, Authors, Type, Submission date}
• The e-teaching.org pattern format uses:
P = {Name/Title, Context, Problem, Forces, Solution, Solution Details, Obstacles, Benefits, Liabilities, Contraindications, Examples, Tools, Related Patterns}

See also figure 88 to compare the fields. A discussion and comparison of description formats for patterns and teaching methods can be found in (Neumann, Derntl & Oberhuemer, 2011).

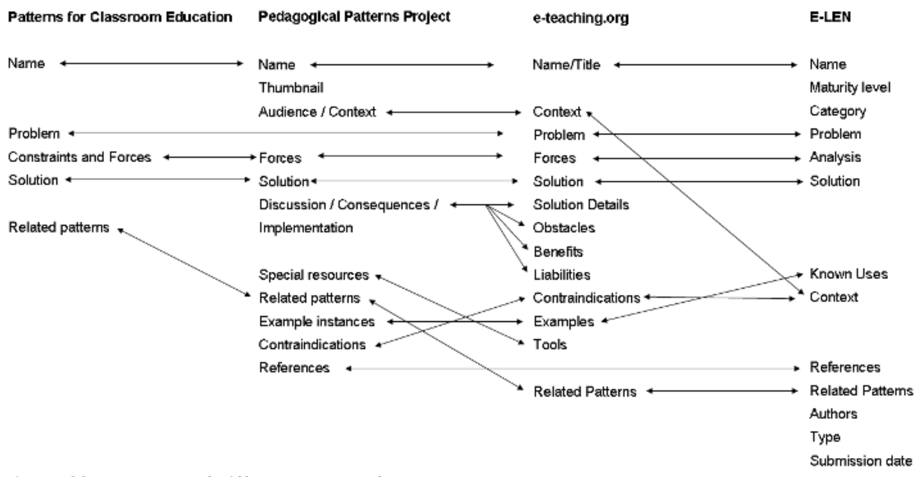


Figure 88. Examples of different pattern formats

To distinguish between the different pattern formats that we used, we will use the labels P2006, EP2007, P2007, EP2008, P2008, EP2010, where

P2006 = {Name, Problem overview, Use when, Forces, Solution summary, Diagram, Solution Details, Implementation, Examples, Rationale, Related Patterns};
EP2007 = {Name, Intent, Problem, Consider this, Therefore, Diagram, Examples, Where can I use this?, Rationale, Roles, States and Interaction};
P2007 = {Name, Intent, Examples, Problem, Solution (with diagrams), Details (with diagrams), What else can I do with this?, Rationale, Related Patterns};
EP2008 = {Name, Intent, Examples, Problem, Forces, Solution (with diagrams), Details (with diagrams), What else can I do?, Rationale, Drawbacks};
EP2010 = {Name, Intent, Context, Problem and Forces, Solution, Examples}.

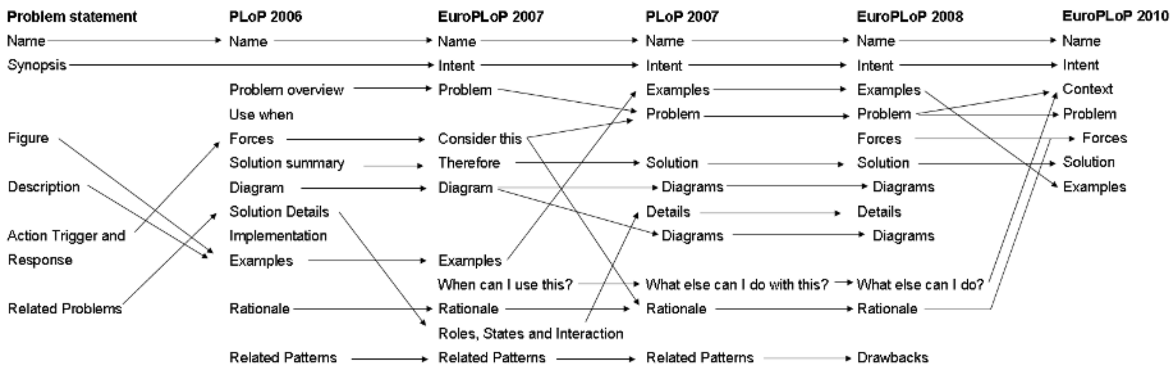


Figure 89. The pattern format to describe the interaction patterns has changed over the years.

4.4.1.3 Which problems to address in written pattern documentations

One of the challenges we encountered was that we only wanted to address the interaction form and its participating elements without providing details about the implementation. Many reviewers at PLoP conferences, however, were mostly interested in implementation details and how the elements relate technically.

But there is a simple reason why the patterns should not provide details about the implementation: the target audience. The written design patterns address end-users – educators and designers – not programmers. The intention of an authoring tool such as moowinx is just that users should not bother about the implementation of the interaction forms. If we review the patterns now, we still find too many details about the implementation.

This is a very illustrative example for mixing up different levels of the problem and solutions as we have discussed in section 2.3.2.6. For educational interactive graphics, we are tackling with pedagogical and human-computer-interface problems. The solutions are configurations of visual elements that react to user input. To implement these structures is a different kind of a problem. Of course there is a solution for this problem as well.

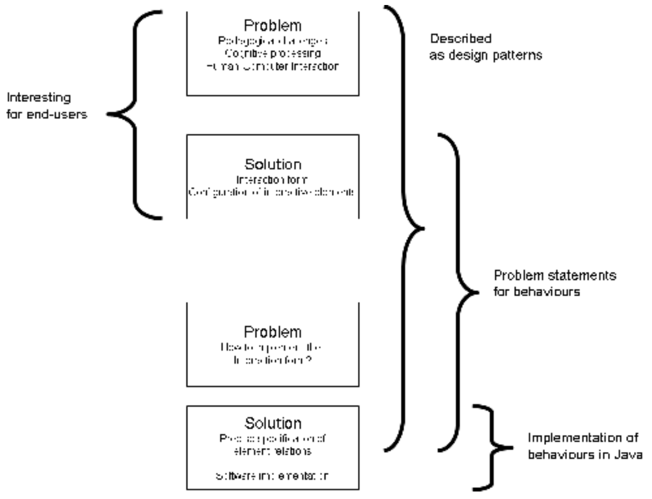

Problem
Pedagogical challenges
Cognitive processing
Human computer interaction
Solution
Interaction form
Configuration of interactive elements
Problem
How to implement the interaction form?
Solution
Precise specification of element relations
Software implementation
Interesting for end-users
Described as design patterns
Problem statements for behaviours
Implementation of behaviours in Java

Figure 90. Different levels of problem and solution.

Figure 90 shows that the design pattern descriptions discuss more details than needed by end-users. The precise specification of relations is only relevant if someone needs to implement the solution by programming.

In fact there are several ways of implementing the same interaction pattern. moowinx offers ready-to-use behaviours to build such solutions. The behaviours themselves are specific implementations in Java that create the interactions. Yet the patterns can be implemented in Flash as well. All of these implementations are solutions of a different level than the emerging structure of interacting elements. There are different configurations on the micro-level of implementation that generate the same interaction form on a macro-level.

The Java implementation of the behaviours is not a pattern on the level of software design. Though each behaviour implements a pattern of interaction, the behaviour implementations are singular software designs. According to pattern ethics, it would be wrong to call such a singular design a pattern. Nevertheless, our pattern descriptions provide many hints for implementations. There are some structural details that always need to be implemented in one way or another. Yet these details are not interesting for end-users. We will see that the implementation details provided vary in the pattern formats; later pattern descriptions contain less implementation detail.

The full descriptions of the patterns can be found in appendix E. The upcoming sections will highlight the changes between the pattern formats and how the description fields cover different information. It is recommended to first scan the pattern descriptions briefly to understand the next sections. Figure 87 and 89 also provide a helpful overview of when the patterns have been documented and how the description fields evolved.

4.4.1.4 Evolution of context descriptions

The problem descriptions (e.g. Switch Problem, Curtain Problem) in my master thesis (Kohls, 2004) aimed at finding required behaviours. They did not have a context statement. In fact, the problems given are not general problems but particular examples; the situation of a particular problem implicitly has a specific context of application. This context, however, was not explicated.

Patterns, on the other hand, require a general context description. Yet, the first documented patterns did not have a field "context". The reason is that the patterns were written for an audience that is not necessarily familiar with the pattern concept. For them it is just important to know in which situations the pattern can be used – its applicability. The context is described under the headlines "Use when", "When can I use this?", "What else can I do with this?" and "What else can I do?".

P2006 starts with the context right after the problem statement whereas EP2007 put the "When can I use this?" at the end of the pattern description. It is much easier to list situations where the pattern can be used if a reader knows already what the pattern is about. P2007 and P2008 change the question to "What else can I do?" The reason is that the examples given at the beginning of the pattern show already general situations in which the solution can be used.

All of the pattern formats P2006, EP2007, P2007 and EP2008 have an enumerative approach that lists different situations in which the patterns apply. The context section of that pattern format states that the pattern can be used in situation A and B and C and … and X.

The pattern format P2010, however, is an experiment to start with the context and describe the situations of applicability in general terms. It is an attempt to generalize over the examples of applications. The examples of applications on the other hand, have moved to the Examples section. This shortens the enumeration of applicable situations because the example section contains already illustrated examples that cover some of the typical situations. For SYNCHRONIZE OBJECT MOTION the illustrated examples include "Move objects automatically in opposite directions", and "Scale, rotate, or delay the motion of two objects" which had been in the context description of EP2007. Otherwise, the enumeration of applications in EP2007 is the same as in the examples of applications in P2010.

For the several patterns of buttons, however, the case is different. The DYNAMIC LABEL (P2006) has the following enumerative context:

Use when:

- A lot of components of the graphic need to be labelled.
- Static labelling makes the graphic too complex or unreadable.
- Draw attention to only one or a limited number of components at a time.
- Provide explanatory verbal information in local contexts.
- Reduce the complexity of a graphic.
- You want to suggest an optimal sequence in which to read the graphic.

The pattern SWITCH BETWEEN OBJECT STATES (P2006) has the following context:

Use when:

- You want to hide or show parts of the graphic only temporarily
- You want to change visual properties such as color or opacity for some visual components to highlight them (or blur them out).
- You want to use the same area of the screen for changing graphical representations while the rest of the screen does not change.
- You want to visually represent On/Off states.

The two patterns DYNAMIC LABEL and SWITCH BETWEEN OBJECT STATES are the predecessors of four variations of buttons presented at EuroPLoP 2008: ON/OFF BUTTON, ROLLOVER BUTTON, INFORMATION DISPLAY, and RADIO BUTTON. While the buttons are technically very similar (they use the same behaviour), their educational applications are very different. Table 13 shows the contexts of application for the different buttons. Note that many of the enumerated situations or cases state what is gained by applying the patterns. They are also expressing the intent.

ON / OFF Button	**Rollover Button**	**Radio Button**	**Information Display**
Switch on and off additional information on the screen.	Highlight related areas or related information on the screen.	Highlight or select one object of a group.	Show a status bar. Provide text information for objects on the screen.
Show pop-up information in small boxes on demand.	Show or highlight all objects of one class or one set.	Direct attention to a particular object.	Show details (e.g. a zoom view) for one of the objects on the screen.
Show the front and the back of an object.	Indicate that an object is activated or that the mouse points at an object that can be activated.	Show explicitly that for a group of objects only one can be active or activated concurrently.	Use the button/hot area objects as menu items to select content.
Switch on and off overlay information for images.	Highlight the bounds of an object, e.g. highlight where the border of a country runs.	Show what to do next or who is next in the row.	Explain components of a larger structure or object.
Switch on and off labels that explain parts of an image.	Dynamically fade in info boxes, windows or text bubbles.	Indicate which object is currently focussed on and handled in a presentation.	Provide information about areas of a map.
Ask a question and show the answer by clicking at the object.	Provide tool tips.	Reduce complexity by hiding or fading all information objects that are currently not needed.	
Reduce complexity by hiding information if not needed.	Show objects in a different state by activating them by the mouse pointer.	Focus on objects step by step in random access.	
Adding information step by step in random order.			

Table 13. Comparison of the "What else can I do?" from different button patterns.

All of our pattern formats start with an intent statement that summarizes the general intent of the pattern. The intent is part of the context in which the pattern can be applied. The enumerations in the "Use when" or "What else can I do?" sections are more specific intents.

Sometimes, the suggested applications also point out a consequence. For example, "Reduce complexity by hiding information if not needed" means that hiding information will, as a positive consequence, reduce the complexity.

4.4.1.5 Evolution of problem descriptions

Most of the pattern formats have separate sections for the core problem and the discussion of forces. Only PLoP2010 uses a single description field "problem"; however, this field has one bold paragraph (core problem) and regular text that discusses the forces under the headline of the problem section. P2007 does not have a forces section but includes forces in the problem statement and the rationale.

The "Curtain Problem" states in its synopsis:

> "Based on the problem of having two curtain halves moved into opposite directions we can handle a class of problems that involve element related movements."

The particular problem is related to a number of similar problems (listed under the "Related Problem Sections") that are later used as examples in the generalized pattern SYNCHRONIZE OBJECT MOTION. The pattern description of SYNCHRONIZE OBJECT MOTION states the problem more general:

> "**How can I demonstrate the correlation of objects in motion?** Cause and effect are difficult to interpret in static illustration because the dimension of time is missing. Correlations in motion cannot be perceived if snapshots only are provided because information about motion paths and speeds between two snapshots is not provided. Arrows can indicate similarity in paths but do not cover variations in speed or time delays. There is no commonly understood semantic for static arrows to map exact information about variation in speed or paused motion. In animations, the user cannot control the motion and has no chance to experiment with differing parameters. Producing such animations is time-consuming if you have to define the motion for each object individually in spite of their dependencies. "

Note that the problem statement starts with a bold "How can I…?" This question addresses the problem of *how* to show correlations of object motions, but it does not cover the deeper problem of *why*, i.e. what happens if we do not explicitly show this correlation? This problem – the actual problem that calls for an interactive rather than a static representation – is discussed in the rest of the problem statement.

The problem statement of the same pattern in P2010 yet sets another focus:

> "A user can control only one object at a time. Even if multiple objects are moved user-controlled at the same time, it is impossible to direct all objects in such a way that the motion paths are identical. Predefining several animation paths for the different objects gets in the way of spontaneous interaction with the graphic."

This problem statement is weak because it does not discuss the pedagogical value; it only discusses a problem of human-computer-interaction - handling several objects at the same time. It focuses on the question: Why that way? This question has actually been neglected in the old pattern version. It is only after we wrote this pattern that we realized the different problem levels: The educational problem, the problem of interaction form, and the problem of implementation. The different problem statements are referring to the same pattern but they are unfolding different aspects of the pattern by asking different questions. It seems that many members of the pattern community still oversee that there are different types of problems. As a result, some patterns focus on one problem type but do not address the other types.

Which problem levels are important depends on the target audience. For example, for our audience the problem of implementation is not relevant because they either use an authoring tool or work together with programmers. We realized this misconception rather late and the problem statements of the button patterns are all very weak. They only ask how to achieve something without asking why to achieve something:

> **ON / OFF Button:**
> How to add or change information dynamically on the screen and let the user decide at which time which information is present?
>
> **Rollover Button:**
> How to add or change information automatically on the screen if the user points on an interesting area? How to provide contextualized information for the area the user currently focuses on?
>
> **Radio Button:**
> How to ensure that only one object out of a group is activated and shows a special visual state?
>
> **Information Display:**
> How to show additional information about objects on the screen without interfering with the content?

None of these problem statements expresses what is problematic. They only address the problem of implementation – the how to…? It is remarkable that this weakness was not pointed out in one of the Writers' Workshops. One of the reasons might be that the "How to…?" statement is still very common in many pattern descriptions. Important aspects of the real educational problem, however, are present in the discussion of forces that follows later, for example "Adding information ad-hoc to provide object specific information avoids split attention effects but the overlay information also hides some parts of the graphic in the background." In retrospection, these forces are a major conflict resolved by ROLLOVER BUTTON and would have been a better problem statement.

4.4.1.6 Evolution of the forces descriptions

All pattern formats except P2007 have an explicit description field "Forces". The pattern format EP2007 has a field "Consider this" which is similar to the description of forces – each bullet point is represented by an icon of competing forces. In addition to the forces section, most pattern formats have a "Rationale" section. This section reasons about why the forces are resolved by the pattern. It also includes references to research results that suggest that the problem is real and something has to be done about it.

The only pattern format without a forces section, P2007, provides reasons about the forces in the rationale section. Moreover, the problem statements in P2007 are more elaborate than in other formats and include many forces.

It is interesting to compare the forces section of SYNCHRONIZE OBJECT MOTION in EP2007 and EP2010 because they are very different yet they reason about the forces of the same pattern:

Forces section in EP2007:

#1: Often, there are implicit motion relations between objects but these only apply if one of the objects is explicitly set into motion. So what triggers the motion of objects? How can you reveal motion relations between objects?

#2: Very often multiple objects are in motion at the same time, but with today's user interfaces you can usually only control one object directly with a (mouse) pointer. So, how can you control many objects simultaneously?

#3: In many cases, multiple objects have to perform identical or similar moves at the same time or with time delays. You could set an animation path for each object individually but this is time consuming and a possible source of error. You may define motion paths according to rules that you have in mind. But if you could express the rules to the machine rather than the results only, ad-hoc creation of new animations would be possible and the definition process becomes more efficient and flexible.

#4: By providing rules, objects can follow other objects, but how can you track the exact path of a moving object? How do you handle variations in speed or direction?

#5: Objects can move into different directions but using the same motion paths. The animation vectors may look totally different but still base on similar paths.

#6: Varying object scales must be respected. Sometimes objects are supposed to move exactly the same paths but they move in differently scaled coordinate systems. Hence, to map the same distance in two different coordinate systems the actual motion on the screen must differ in the step size measured in pixels.

Forces section in EP2010:

Predefined paths will lead to deterministic animation that is not interactive at all. Thus, experimenting and exploring what happens if one object is moved would not be possible. Defining animations paths is very time consuming as well and it forbids the ad-hoc creation of new animations and take control of the interactive graphic. (#7)

Simply grouping the objects that should move together could only help if all objects are supposed to move in exactly the same way. Such large groups often complicate the access to other objects on the screen and you accept that each element of the group can cause motion (#8). Often, there are implicit motion relations between objects but these do only apply if one of the objects is explicitly set into motion. So, what triggers the motion of specific objects? How can you reveal motion relations between objects? How do you handle variations in speed, directions or time delays? What can you do if the objects that are supposed to move are on backgrounds that use different scales? (#10)

EP2007 is much more detailed. It also highlights the conflict between forces by including "but", "however" etc. This connection of contradicting or conflicting forces is also found in all of the button patterns in EP2008.

EP2010 on the other hand tries to reason about the forces in a flow of text rather than an enumeration of forces as bullet points. The forces section #7 (EP2010) is a rewording of #3 (EP2007). The forces section #10 (EP2010) summarizes the forces #1, #6, #5, and #6 from EP2007 and restates them as questions.

EP2010 also takes new developments in account. The force #2 "with today's user interfaces you can usually only control one object directly with a (mouse) pointer" is no longer present because there are multitouch devices available. Forces section #8 about grouping objects is new in EP2010 and has no equivalent in EP2007. It reflects the experience that similar things can be done with grouping features but less conviently and less flexible.

4.4.1.7 Evolution of solution descriptions

The Switch Problem and the Curtain Problem both offer a brief description of the elements that are involved in implementing the interaction described in the problem examples. This description is found under a headline "Action Trigger and Response".

For the Curtain Problem, the description is:

> To implement a movement relation the user can add a follow-move behaviour to the element that should be guided. One of the behaviour's parameters is the leading element. Other parameters specify the follow type, e.g. directly, delayed, opposite, rotated, or an acceleration factor. […] The leading element can be set to dragable so that the user can manually move the element later on the slide. In the curtain example it make sense to apply the same behaviour to both elements because it does not matter at which side you open the curtain. In other examples the leading movement is restricted to one element.

The EP2007 version of SYNCHRONIZE OBJECT MOTION summarizes the solution even briefer and introduces names for the different roles of the participating objects:

> Therefore:
> Relate *Guide* and *Fellow* objects that synchronize their motion automatically. Each move of a *Guide* object is captured as a vector and applied to all *Fellow* objects. The motion vector can be identical or transformed, i.e. scaled, rotated or delayed.

The solution is supported by a diagram that shows the different transformations and a discussion of Roles, States and Interaction. In this discussion, each of the roles (Guide and Fellow) and their relations are exactly described. The discussion also includes details about the transformations along with specific examples:

> "To respect different scales and speeds of objects, the motion vector of the *Guide* and its *Fellows* can be scaled by a scale factor. The scale factor defines the relative step size between object A's and B's motion. For example, let the scale factor be 2 and object A moves 15 pixels. Then object B moves automatically 30 pixels. On the other hand, if B moves 50 pixels, then A moves 25 pixels. "

The notion of a motion vector is an implementation detail that is only important to understand if one wants to implement the pattern by programming.

The content of the solution description of the same pattern in its EP2010 has not changed essentially. However, some statements have been rephrased and some examples have been taken away in the EP2010 version to make the elaboration shorter. The following examples for scaling are found in EP2007 but not in EP2010:

> Varying scale factors can be used for different object relations if you define multiple *Guide-Fellow* relations. The semantically meaning of the scale factor depends on the represented objects. A scale factor 2 can express that object B is two times faster than A. It can also express that A and B move the same distance but in differently scaled coordinate systems, i.e. the coordinate system of B is two times larger than the coordinate system of A.

Thus, EP2010 has the same general discussion but fewer examples. It covers the same design space and information content (e.g. number of cases that fall under the pattern) but provides fewer illustrations. We can compare this to the description of a common category such as Dog. The concept of a Dog does not change whether we provide three or ten examples. However, our understanding of what kinds of Dogs there are improves if we see more examples.

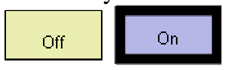
A very interesting evolution of solution descriptions can be observed for the button patterns. In P2006, the pattern DYNAMIC LABEL discusses in its solution details some of the variants, e.g. to click on a button, rollover a button, and only allowing one button to be switched ON are discussed as variants of the same solution:

> [ROLLOVER:] …The simplest way to activate a dynamic label is a ROLL-OVER with the mouse. The user finds out very quickly where labels are by moving the mouse over areas he is interested in…
>
> [ON/OFF BUTTON:] …To keep a label permanently visible you can use a SWITCH BETWEEN OBJECT STATES. If the user clicks at the relevant area, the label appears. To let it disappear the user has to switch it explicitly off. This could be done by clicking at the label or at the relevant area. This way more than one label can be displayed at the same time…

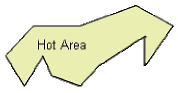
> [RADIO BUTTON:] …Another option is to switch the label off as soon as another label is selected. This allows only one label at a time. Showing only one label can be useful in presentations to direct the audience's attention and HIGHLIGHT INFORMATION. If there are many dynamic labels or each label is large in size then multiple viewed labels would overlap. Automatically switching off one label as soon as another is switched on reduces the mouse clicks to move on from one label to another.

In EP2008 the variants are described as different patterns. Thus, the solution only needs to discuss one of the alternatives. The EP2008 format does not have special "Diagram" sections but mixes the description of the core solution with illustrations:

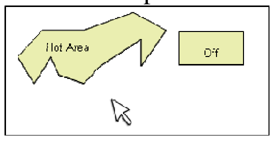
> **Solution of Rollover Pattern:**
> For a visual object define two visual states ON and OFF by having some visual properties set differently:



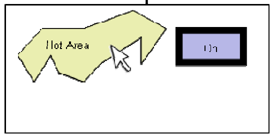
> Define a hot area that is sensitive to the mouse pointer:



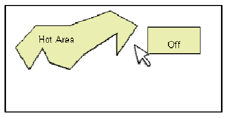
> If the mouse pointer is outside the hot area, show the OFF state:



> If the mouse pointer is within the hot area, show the ON state:



> On exiting the hot area, return to the OFF state:



The solution in P2006 is more compact because it subsumes solution variations in one description. The four different solution descriptions in EP2008 have many redundancies because they all describe the same solution form with only minor variations (such as mouse click vs. rollover). However, these minor variations have a high impact. The application context, the forces and consequences are quite different. That these small variations change the meaning significantly is also reflected in the discussion sections. For example, the details of Rollover Button illustrate some general applications of the solution as well as the consequences:

> Because the ON state is only temporarily for an object, a roll over can indicate that an object or an area is activated at the moment. It can highlight which object is focussed by the mouse pointer. If objects on the screen offer an affordance for further operations (i.e. clicking at an object or drag the object), the highlighting indicates on which object the mouse pointer would operate.
>
> The transient character of rollover buttons can be useful to highlight related objects on the screen. For example, by moving the mouse pointer over one object of a set, all members of the set could be highlighted.

4.4.1.8 Core solution and solution details

The solution statements are always split into the core solution and solution details. The core solutions introduce roles and describe what happens. The solution details explain how the relations work exactly.

For example, the solution summary in DYNAMIC LABELS is: “Dynamic labels appear on the screen only when needed.” The solutions details provide the information how the appearance is triggered and what is meant by “when needed”, i.e. to highlight information or reduce the amount of information on the screen.

Likewise, the core solution of SYNCHRONIZE OBJECT MOTION introduces a *Guide* and *Fellow* object that synchronize their motion automatically. The solution details explore how the synchronization is performed by capturing and transforming a motion vector.

P2006 splits the solution details into the section “Solution Details” and “Implementation”. The implementation does not cover programming details but suggests the implementation in authoring tools or presentation software:

> Most authoring environments allow you to dynamically show or hide components and you can implement the solution using a SWITCH BETWEEN OBJECT STATES. On the contrary, presentation software usually does not provide features to switch on and off parts of a graphic. However, most presentation tools allow navigation and hyperlinking. If you run through a predefined sequence of dynamic labelling, you can just multiply the same graphic on a sequence of slides and use only one label per slide. The standard functions to step forward/back will guide the user from one label to the next. If each label should be accessible at any time you must provide hyperlinks on all slides showing the graphic. For the hyperlinks, use invisible objects (with transparent opacity) and place them at the areas where a mouse click should pop up the label. This method, of course, is only practicable if the number of labels is small.

Some of the implementation suggestions are questionable. While it is possible to emulate the dynamic labelling of objects or show and hide objects via hyperlinks, this method is not very practical. In fact, we have never seen such an implementation. So, this implementation does not report a proven solution for presentation software. While there is evidence for the pattern there is no evidence that the implementation suggested is useful. It would work but it is not very practical. What is worse, most presentation tools do offer better ways to switch on and off elements. For example, PowerPoint directly supports this operation though it is not very well known. In moowinx the implementation of dynamic labelling can be done instantly with one of the behaviours or wizards. The implementation section in P2006 was an attempt to show that the interaction could be implemented in other environments as well. However, taken into account that hyperlinks are not known to be used for labelling and that there are other options, we have to falsify the suggested implementation as a good solution. However, that does not falsify the solution of dynamic labelling. Again, we are encountering two different levels of the solution. One level is the configuration of interaction elements (still valid); the other is its technical implementation by hyperlinks (not appropriate).

While DYNAMIC LABELS is still a valid view on the solution, the variations of different buttons in EP2008 provide more details about the solutions and their exact interaction. EP2008 does not provide any suggestions for implementation. It only explains the elements that are always involved. For example, there always needs to be an area that triggers the switching of states. Yet there are different ways of defining this area. In fact, different authoring tools and programming languages will lead to very different implementations. While there is a pattern of interaction (macro-state) there is no pattern in technical implementation (different micro-states).

EP2007 describes the solution details in “Roles, States and Interaction”. That section provides details about the roles of participating objects and how these objects interact. The different ways of transforming the motion vector of Synchronize Object Motion also discusses variations of the solution. The pattern formats P2007 and P2008 focus on the variations of the solution in their “Details” section. The core solution statement is already detailed and includes the full description of interaction and diagrams for illustrations. In this case, the “Details” sections are not clarifying the interaction form but show the variations and their consequences.

The pattern format EP2010 has no “Solutions Details” section. Rather it has one solution section that highlights the core solution in bold font and continues with the solution details in the same section. Unfortunately, the patterns no longer include the diagrams, making them less accessible. The EP2010 pattern format was an experiment whether the patterns could be described briefer and with fewer sections. From the feedback of the shepherding process and the Writers’ Workshop, however, it became clear that the shortening made the patterns less mature.

4.4.1.9 Diagrams:

All of the patterns formats – except EP2010 – include diagrams to illustrate the solution. The pattern formats P2006 and EP2007 had explicit headlines for the diagrams. P2007 and EP2008 include the diagrams in the solution description, integrating text and illustrations into one flow.

The illustrations are mappings of the dynamic interactions on a computer display. For each significant change, a frame illustrates the different states and different locations of objects.

The diagram of Dynamic Labels (P2006) is different to the other diagrams for two reasons. First, it does not show a sequence of different states but a single image. Second, it does not show the interaction of abstract object but an actual example.

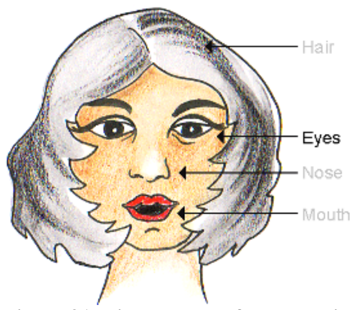


Figure 91. Diagram used for Dynamic Labels description.

SWITCHING BETWEEN OBJECT STATES (P2006) follows the same strategy as the other pattern formats and represents images of significant changes using abstract objects.

That the diagrams show abstract objects does not mean that they represent an abstraction of the pattern. One can use the abstract objects (such as a circle or rectangle) in an actual instance of the pattern. For example, we can define motion synchronization between two circles on the screen. This is important evidence for our claim that models can be real exemplars of a pattern (see section 3.2.7.4). If we synchronize circles, we synchronize abstract objects. Yet the interaction is as concrete and specific as any of the other examples. The benefit of using abstract objects in the diagrams is that these meaningless objects do not distract us from the meaning of the pattern. If one circle follows another we concentrate on the mechanism that lets one circle follow the other. Examples put the mechanism into "real world" situations and such situations are laden with further meaning that could lead to misinterpretations. A good pattern description should contain both an abstract representation and illustrating examples.

4.4.1.10 Examples sections

The diagrams show the concrete interaction but abstract from any particular situation where the interaction is useful. To understand where we can use the interaction, we need to understand the situations where they apply. Therefore, each pattern description includes several examples (at least three) to illustrate very different situations in which the same interaction pattern can be used.

It is interesting to note that the examples do not only illustrate the interactions but specific situations and contexts as well. This explains why the lists of applicable contexts in EP2007 and P2007 have moved to the "Examples" section in EP2010. The lists of applicable contexts were in fact just examples of the general context. The pattern format EP2010 tries to describe the context in more general terms; thus the context examples have moved to the example section. The example section in EP2010 now contains both a list of examples and detailed examples that are supported by illustrations.

That the examples implicitly describe contexts of applicability is also reflected in renaming the context description from "When can I use this?" (P2006, EP2007) to "What else can I do with this?" (P2007, EP2008). The illustrated examples sections in P2007 and EP2008 have moved to the beginning of the pattern descriptions. Thus, some contexts of applications are shown at the beginning already. The question "What else can I do with this?" asks which additional contexts, apart from the given examples, are there?

The reason why we moved the examples section to the beginning of the pattern descriptions was to keep the whole discussion of the pattern less abstract. When the examples were at the end, the discussion of problem, forces and the core solution had been very abstract. Readers might have wondered: where is this going? To start with the examples had several benefits. It allowed an inductive approach, starting with specific instances and extracting the general interaction form. Starting with examples is also more motivating because readers learn at the beginning what the pattern is all about. It also means that the pattern starts with the context (or examples of applicable contexts) instead of a problem. Since problem and forces are always situated, their understanding depends on understanding the context first. That is why the EP2010 starts with a context description right away. Unfortunately, in EP2010 the examples moved back to the end of the pattern description – that means the pattern again starts with the abstract ideas and only offers specific examples at the end.

4.4.1.11 Evolution of the consequences section

None of the pattern formats has an explicit section to discuss the consequences. However, the "Details" section of P2007 and EP2008 discuss the consequences for each of the variations of the pattern. Another section that reasons about the consequences is the Rationale given for each pattern.

The only pattern format that discusses negative consequences in a separate description field is EP2008. EP2008 has a description field "Drawback". The goal is to support users in their decision which of the button patterns they should choose. Each of the patterns in EP2008 offers a variant of a more general pattern. If there are variations to choose from one has to make a decision for one of the competing variants based on the benefits and drawbacks. While each pattern does this for its own variations in the details section, the "Drawback" section helps to compare the alternative patterns directly. For example, table 14 compares the drawbacks between an ON/OFF button and a Rollover button:

ON / OFF Button	**Rollover Button**
Each button can only set two states, ON and OFF. Many objects in the real world, however, offer more than two interesting visual states. To assign multiple visuals states to an object, an INFORMATION DISPLAY can be used. The user has to explicitly turn off an element to deactivate it. In particular, if too many elements are activated simultaneously, an image can become easily overloaded. RADIO BUTTONS and ROLLOVER buttons automatically turn OFF an activated element for the price of having only one element activated at a time. The changeable graphic elements have to be equipped with their own affordances in such a way that the user can understand where s/he can click.	Using the mouse pointer to activate elements of a graphic means that the pointer cannot be used for anything else while activating the specific information. To let information pop out for a longer time period, a RADIO BUTTON is an alternative that also takes care of only having one information unit activated at a time. A RADIO BUTTON, however, is less intuitive as it requires a more explicit user input (mouse clicks). Some interactive displays, such as touch monitors and some interactive whiteboards, do not recognise mouse motion without pressing the surface. Hence a rollover (which occurs on moving the mouse without pressing the mouse button) will not work. As an option, one can design an ON/OFF button that responds to pressing the mouse button (instead of mouse entering) and releasing the mouse button (instead of mouse exiting). It is not possible to select multiple elements simultaneously (e.g. to compare representations or reduce the number of objects). Since each added information is automatically discarded if the focus is lost, a ROLLOVER can not be used to create complex illustrations step-by-step. To implement such behaviour, use an ON/OFF BUTTON instead.

Table 14. Drawbacks of ON/OFF button and Rollover button compared

The discussion of the drawback also provides references to the alternative patterns that do not have that drawback. So, if one of the drawbacks is a serious issue in a given context, one can move on to one of the suggested alternative patterns.

4.4.2 Reflection on the evolution of pattern descriptions of interactive graphics

The evolution of the patterns (their format and content) has been driven by several factors:
- Feedback from shepherds and participants of Writers' Workshops
- Deeper understanding of the patterns themselves
- Results from our experiments about the induction of patterns
- Deeper understanding of the pattern concept from the path metaphor, theoretical framework and discussions in Writers' Workshops
- Familiarizing with a larger body of pattern literature and different styles

We will discuss the most important factors in the next sections.

4.4.2.1 Comments from shepherds and Writers' Workshops

Comments from shepherds and participants of Writers' Workshops have had a large impact on the restructuring and clarification of the patterns.

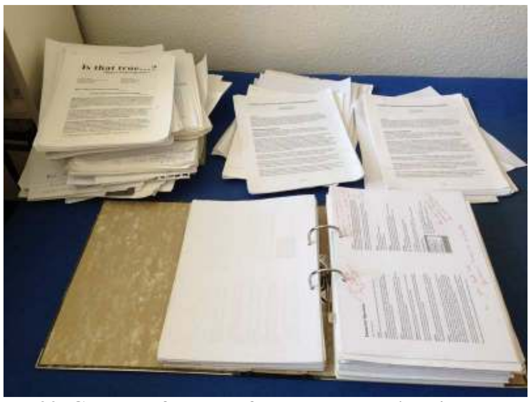


Figure 92. Collected feedback from several Writers' Workshops.

Both shepherds and participants of Writers' Workshops asked for clarification of the target group. This led to a reflection about the intended audience and a decision to include less technical information. The reason is that the patterns address end-users rather than developers.

Another frequent comment was to have specific consequences sections. While the solution discussions should have covered these, a dedicated section would have been better. EP2008 explicitly highlights the drawbacks. The description of patterns for online trainings (see next chapter) also makes the consequences more explicit by naming obstacles, benefits and liabilities.

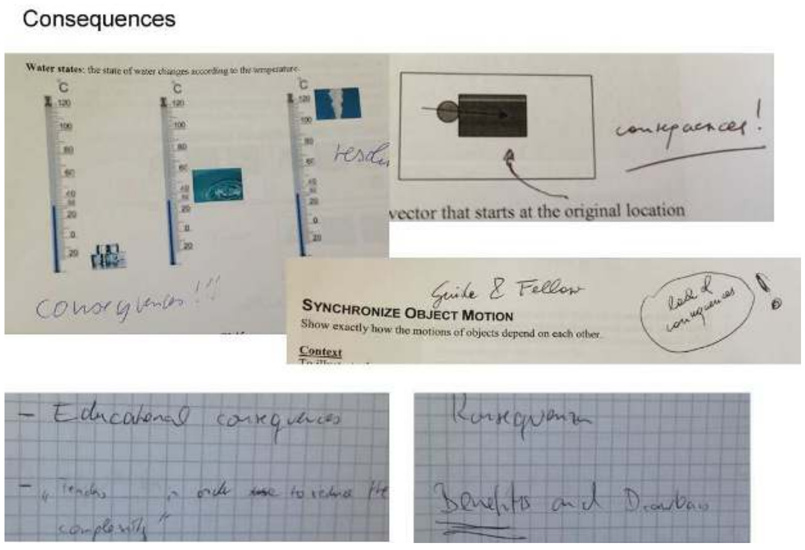
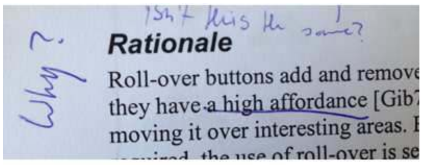

Consequences
Water states: the state of water changes according to the temperature
consequences!
vector that starts at the original location
Guide & Fellow
SYNCHRONIZE OBJECT MOTION
Show exactly how the motions of objects depend on each other.
Context
- Educational consequences
Konsequenzen
Benefits and Drawbacks

Figure 93. Comments from Writers' Workshops on consequences.

There have been several comments about the problem and the forces and it was suggested to make the problem statement more “problematic”. Instead of just stating the intent or purpose, the problem should show what is bad or critical if the pattern (or an alternative solution) is not applied. Problem statements beginning with “How to…” have been criticized because they do not describe the original problem that is being solved but only address the problem of implementation. Since the problem statement and the forces address problems on different levels, both had been combined in P2007 into one section. However, the feedback in the Writers’ Workshop was that separate problem and forces sections would be better. Hence, EP2010 has the two sections separated again using explicit headlines.


Isn't this the same?

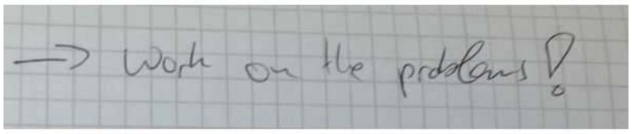
Why?

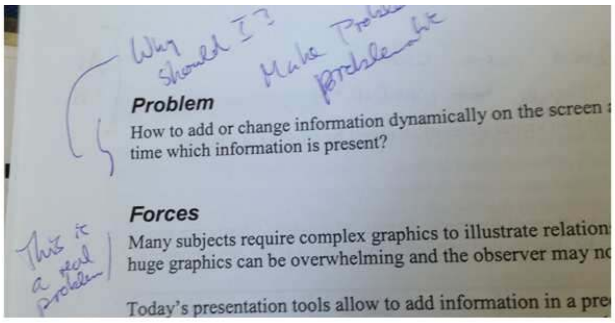
Rationale

Roll-over buttons add and remove

they have a high affordance [Gib7

moving it over interesting areas.


→ Work on the problems!


Why should I?

Make Problem problematic

Problem

How to add or change information dynamically on the screen

time which information is present?

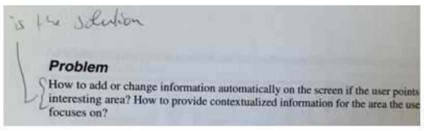
This is a real problem

Forces

Many subjects require complex graphics to illustrate relation

huge graphics can be overwhelming and the observer may n

Today’s presentation tools allow to add information in a pre

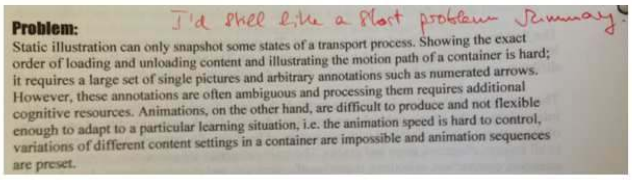

is the solution

Problem

How to add or change information automatically on the screen if the user points

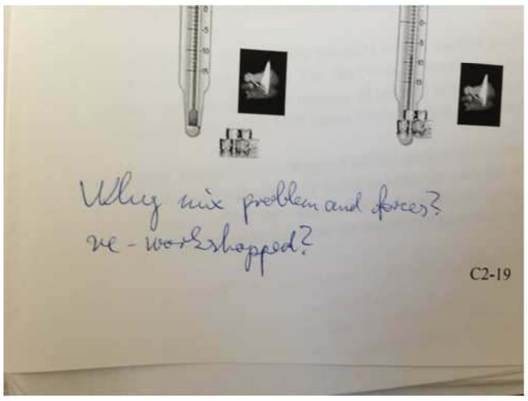
interesting area? How to provide contextualized information for the area the use

focuses on?


Problem:

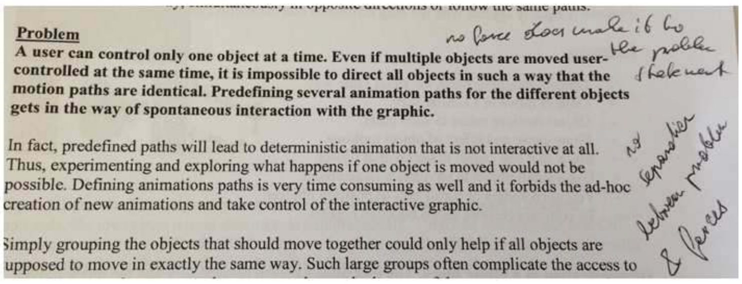
I'd still like a short problem summary

Static illustration can only snapshot some states of a transport process. Showing the exact order of loading and unloading content and illustrating the motion path of a container is hard; it requires a large set of single pictures and arbitrary annotations such as numerated arrows. However, these annotations are often ambiguous and processing them requires additional cognitive resources. Animations, on the other hand, are difficult to produce and not flexible enough to adapt to a particular learning situation, i.e. the animation speed is hard to control, variations of different content settings in a container are impossible and animation sequences are preset.


Why mix problem and forces?

re-workshopped?

C2-19


Problem

no force does make it to the problem statement

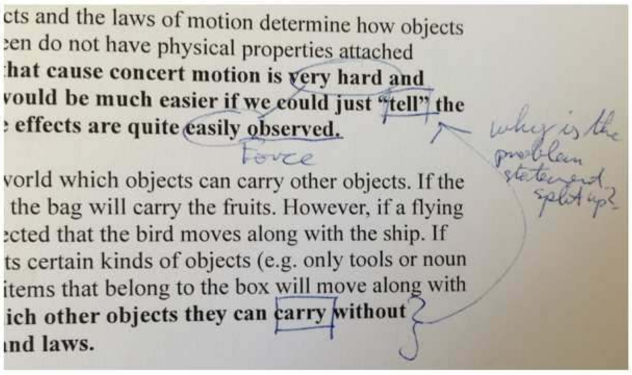
A user can control only one object at a time. Even if multiple objects are moved user-controlled at the same time, it is impossible to direct all objects in such a way that the motion paths are identical. Predefining several animation paths for the different objects gets in the way of spontaneous interaction with the graphic.

In fact, predefined paths will lead to deterministic animation that is not interactive at all. Thus, experimenting and exploring what happens if one object is moved would not be possible. Defining animations paths is very time consuming as well and it forbids the ad-hoc creation of new animations and take control of the interactive graphic.

no separation between problem & forces

Simply grouping the objects that should move together could only help if all objects are

upposed to move in exactly the same way. Such large groups often complicate the access to


cts and the laws of motion determine how objects

en do not have physical properties attached

hat cause concert motion is very hard and

vould be much easier if we could just “tell” the

effects are quite easily observed.

Force

why is the problem statement split up?

vorld which objects can carry other objects. If the

the bag will carry the fruits. However, if a flying

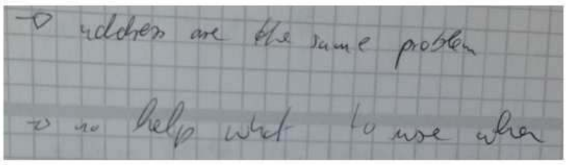
cted that the bird moves along with the ship. If

ts certain kinds of objects (e.g. only tools or noun

items that belong to the box will move along with

ich other objects they can carry without

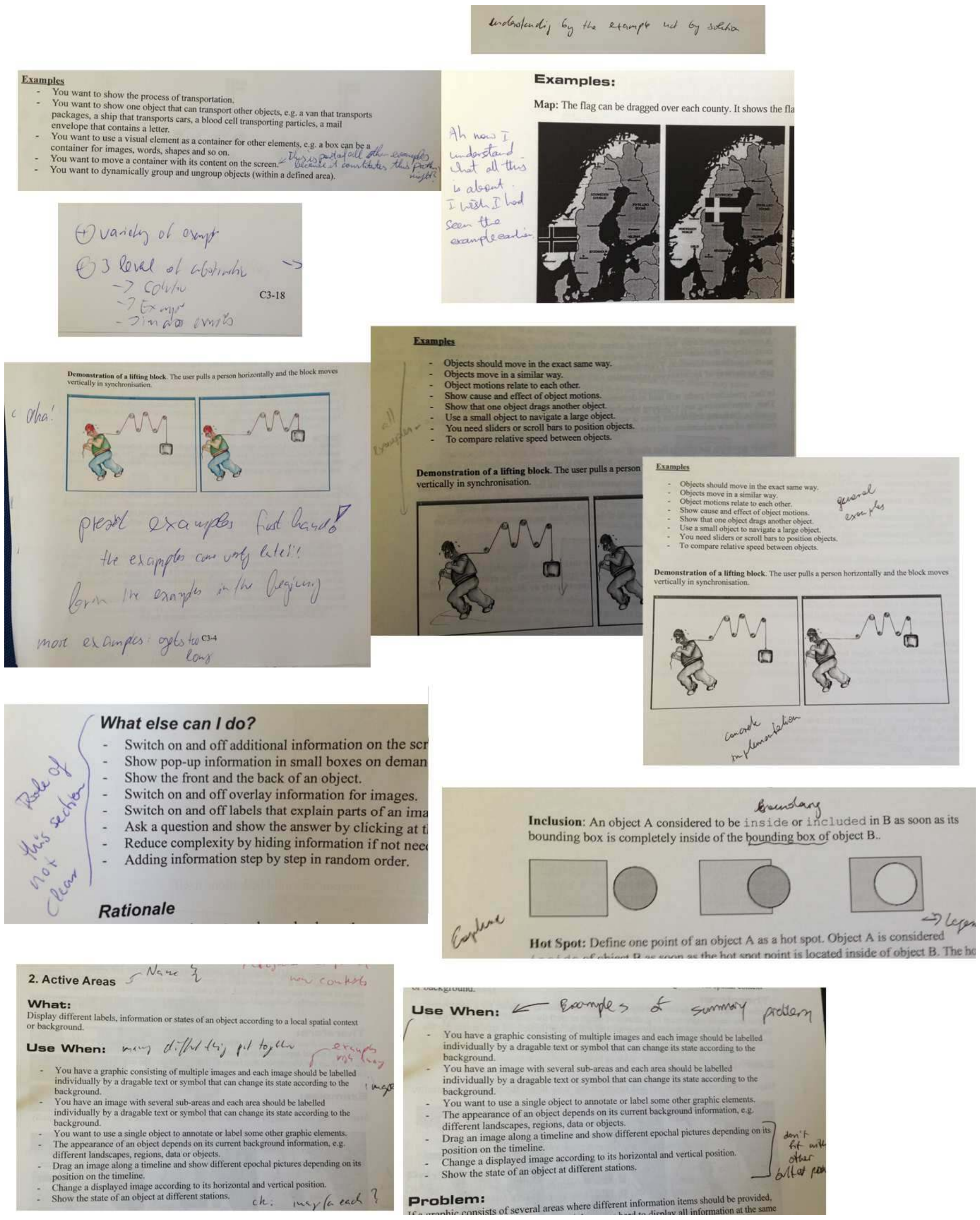
ınd laws.


→ rddhen are the same problem

→ no help what to use when

Figure 94. Comments from Writers’ Workshops on problem and forces.

The relations between the general solution, illustrative examples and examples of further application have been discussed in several workshops. Many participants had commented that they understood the patterns only after reading the examples. Therefore, the examples moved to the beginning of the pattern description in P2007 and P2008. The solution section showed the form in an abstract way whereas the illustrated examples showed the form in very particular ways. The enumeration of further examples was perceived differently. When it was named "Use when", participants commented that the section showed examples. When it was named "Examples" participants pointed out that the enumeration showed rather general situations of where the patterns are applicable (context). As a result, EP2010 tries to balance the two approaches by providing specific illustrated examples and generic examples in the context statement.



understanding by the example not by solution

Examples

- You want to show the process of transportation.
- You want to show one object that can transport other objects, e.g. a van that transports packages, a ship that transports cars, a blood cell transporting particles, a mail envelope that contains a letter.
- You want to use a visual element as a container for other elements, e.g. a box can be a container for images, words, shapes and so on.
- You want to move a container with its content on the screen. ← this is part of all other examples because it constitutes this pattern might?
- You want to dynamically group and ungroup objects (within a defined area).

Examples:

Map: The flag can be dragged over each county. It shows the fla

Ah now I understand what all this is about. I wish I had seen the example earlier.

(+) variety of exampl

(–) 3 level of abstraction →
→ solution
→ Exampl
– Similar events

C3-18

Demonstration of a lifting block. The user pulls a person horizontally and the block moves vertically in synchronisation.

aha!

present examples first hand!

the examples come only later!!

bring the examples in the beginning

most examples: ogles too long

C3-4

Examples

- Objects should move in the exact same way.
- Objects move in a similar way.
- Object motions relate to each other.
- Show cause and effect of object motions.
- Show that one object drags another object.
- Use a small object to navigate a large object.
- You need sliders or scroll bars to position objects.
- To compare relative speed between objects.

all Examples

Demonstration of a lifting block. The user pulls a person vertically in synchronisation.

Examples

- Objects should move in the exact same way.
- Objects move in a similar way.
- Object motions relate to each other.
- Show cause and effect of object motions.
- Show that one object drags another object.
- Use a small object to navigate a large object.
- You need sliders or scroll bars to position objects.
- To compare relative speed between objects.

general examples

Demonstration of a lifting block. The user pulls a person horizontally and the block moves vertically in synchronisation.

concrete implementation

What else can I do?

- Switch on and off additional information on the scr
- Show pop-up information in small boxes on deman
- Show the front and the back of an object.
- Switch on and off overlay information for images.
- Switch on and off labels that explain parts of an ima
- Ask a question and show the answer by clicking at t
- Reduce complexity by hiding information if not nee
- Adding information step by step in random order.

Role of this section not clear

Rationale

boundary

Inclusion: An object A considered to be inside or included in B as soon as its bounding box is completely inside of the bounding box of object B..

Explain

→ legen

Hot Spot: Define one point of an object A as a hot spot. Object A is considered

2. Active Areas ← Name ?

new contexts

What:
Display different labels, information or states of an object according to a local spatial context or background.

Use When: many different things put together

examples not many

- You have a graphic consisting of multiple images and each image should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You have an image with several sub-areas and each area should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You want to use a single object to annotate or label some other graphic elements.
- The appearance of an object depends on its current background information, e.g. different landscapes, regions, data or objects.
- Drag an image along a timeline and show different epochal pictures depending on its position on the timeline.
- Change a displayed image according to its horizontal and vertical position.
- Show the state of an object at different stations.

1 image

ch: image for each ?

Use When: ← Examples of summary problem

- You have a graphic consisting of multiple images and each image should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You have an image with several sub-areas and each area should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You want to use a single object to annotate or label some other graphic elements.
- The appearance of an object depends on its current background information, e.g. different landscapes, regions, data or objects.
- Drag an image along a timeline and show different epochal pictures depending on its position on the timeline.
- Change a displayed image according to its horizontal and vertical position.
- Show the state of an object at different stations.

don't fit with other bullet points

Problem:
If a graphic consists of several areas where different information items should be provided,

Figure 95. Comments from Writers' Workshops on examples, solutions and contexts.

4.4.2.2 Deeper understanding of the patterns

The patterns have also evolved as the authors got a deeper understanding of the patterns themselves. Even if one knows a pattern very well, more experience can change the view on the pattern. This is similar to knowing how to drive a car and still learning about it everyday. Deeper knowledge of the patterns is reflected in taking new developments into account (multi-touch vs. single-touch devices), learning what does not work (for example the wrong implementation suggestions for PowerPoint given in P2006) or seeing that small variations matter more than anticipated before (splitting DYNAMIC LABELS into several sub-patterns). In this case the evolution of the pattern content is based on more experience and taking more cases into account.

The experiments about the induction of patterns also suggested some restructuring of the original patterns. By differentiating between the solution forms it became clear that the various types matched different contexts and raised different consequences.

The evolution of the patterns was also based on a better understanding of the pattern concept and seeing many different pattern styles used in Writers' Workshops. The discussions in Writers' Workshops do not only provide feedback for the author's paper but illuminate many of the deeper concepts of patterns.

4.4.2.3 Learnings from the path metaphor

In particular the path metaphor has helped to understand and visualize which key questions should be answered by each of the sections (Kohls, 2012c):

> The key to describe the **context** is to ask when-questions. When can this pattern be applied? When should it not be applied? Who can use the pattern? In which situations can the pattern be used? Which other patterns have led to this context?
>
> The key to describe the **problem and forces** is to ask why-questions. Why do we have to follow this specific path? Why do we have to go there? Why do we need this specific form? Why can't we do another thing instead? Each single force gives another answer to such why-questions. A force explains the cause for a specific design decision by giving the "because" to the "why".
>
> The key to describe the **solution** is to ask what-, how- and who-questions. What is the general structure of the solutions? How can I generate the solution structure? How do the parts relate and interact? Who is participating in the solution? What are variations of the solution?
>
> The key questions of the **consequences** section are: Where are we now and what should we do next? What are the benefits, costs, drawbacks, tradeoffs and liabilities of the solution? Which forces have been resolved or weakened? Which forces remain or have been newly introduced? What needs to be done next and which other patterns could support this solution?

The path metaphor was originally developed to explain the pattern concept in common sense terms to potential authors who are not familiar with the pattern approach. The next section will discuss patterns for online trainings. These patterns have been developed with the path metaphor in mind and take many of the insights gained from evolving the patterns for interactive graphics into account. Many remarks in Writers' Workshops expressed concern about the missing "Consequences" section. The path metaphor helped to provide a differentiated view on the consequences: every hiker knows that once a path is chosen there are obstacles (such as a small canyon), benefits (such as scenic views), liabilities (such as carrying enough water) and contraindications (such as not starting too late in the evening).

4.4.3 Describing a pattern language for online trainings

4.4.3.1 Background

The patterns for online trainings (Kohls, 2009a) form a pattern language and are part of e-teaching.org, an online portal as an information resource for academic teachers who want to integrate digital media into everyday teaching.

The portal e-teaching.org was launched in 2003 and has become an established information resource for e-learning activities at universities in German speaking countries. It provides information about didactical, organizational and technological aspects of e-learning. The aim is to support local e-teaching initiatives in higher education. Analyses of log file data reveal that half of all German universities regularly visited the portal, each day the portal has several thousand visitors. e-teaching.org offers practicable and rich information about using digital media in teaching and about how to foster this use.

The content is structured along the lines of the user's individual motivation and qualifications, e.g. media technologies, teaching scenarios, good practice examples or didactical design information. In 2009, e-teaching.org delivered a special on e-learning patterns. The special included case studies and introduction texts about patterns. There is a dedicated section about design patterns (Instructional Design -> Concepts -> Design Patterns). Moreover, the section teaching scenarios is extended with new patterns.

4.4.3.2 Mining ground

The patterns ground in experiences of online training events of the German information portal e-teaching.org. Between spring 2006 and spring 2009 there have been 14 online trainings, hosted by the author and guest trainers. There are usually between 30-60 participants attending the live event. Each event was recorded. The trainings are open educational resources and can be accessed at[33]:
http://www.e-teaching.org/community/communityevents/schulung/

Besides our own experiences, we discussed best practices of using the conference system Adobe Connect at a user meeting. Many of the patterns we applied have been in use by others independently, i.e. INVISIBLE CO-HOST, CONTROL MONITOR and TEST RUNS. Though we have gathered our experiences with a particular conferencing system, they are not limited to that system. We have tested other systems as well and their functions are quite comparable. The pattern language should be useful for any of the standard web conference systems. The complete language was discussed in a Writers' Workshop of EuroPLoP 2009 (Kohls, 2009a). A revised version can be found in appendix F.

4.4.3.3 The e-teaching.org pattern format

The e-teaching.org pattern format is:

P = {Name/Title, Context, Problem, Forces, Solution, Solution Details, Obstacles, Benefits, Liabilities, Contraindications, Examples, Tools, Related Patterns}.

Each of the description fields is guided by a question (Kohls, 2009b):

Context: In which situation or in which environment is this pattern actually useful?
Problem: Which core problem is addressed by the solution form?
Forces: Which forces are present in the context?
Solution: What is the general form of the solution?
Solution Details: How to implement the solution? Which variations exist?
Obstacles: What is critical to make the solution a success?
Benefits: Which value is gained by using the solution?
Liabilities: Which liabilities have to be accepted?
Contraindications: What are typical situations in which the pattern should not be applied?
Examples: Which cases or instances of the pattern could be listed?
Tools: Which tools can be useful to implement the pattern?

While the contexts describes typical situation in which the pattern is applicable, the description field "Contraindications" describes situations where the pattern should not be applied. This has impact on the logical and empirical content of the pattern as discussed in section 3.4.5. The context can explicate both when and when not to use the pattern.

The consequences of the patterns are discussed in much more detail than for the patterns of interactive graphics. The descriptions of obstacles, liabilities and benefits all discuss consequences, i.e. the field of forces after applying the pattern. The benefits discuss which conflicts of forces have been resolved. The liabilities discuss which conflicts remain or which new conflicts arise. The obstacle section discusses particular forces that arise as a consequence of applying the pattern; it also offers an instant resolution to these forces or points to sub patterns that can resolve the forces.

[33] The trainings are in German.

4.4.3.4 Relations between patterns

The kinds of references between patterns show major differences between the patterns of interactive graphics and the patterns of online trainings. The patterns of interactive graphics all have a section *Related Patterns*. These sections, however, point to patterns that solve the same problem in a different way or belong to the same super ordinate category. The patterns for online training are interconnected in a different way: the entry pattern ONLINE TRAINING *builds* on other patterns; hence it refers to them in its solution description. An ONLINE TRAINING is composed of the interplay of other patterns. The relation between the patterns is that they work together, e.g. a REHEARSAL tests PREPARED EXAMPLES and the TRAINING OUTLINE. CONTROL MONITOR and INVISIBLE CO-HOST support the success of an ONLINE TRAINING. The patterns also help to solve problems that occur in the context of higher level patterns.

Let us consider how the patterns for online training cooperate. First, we can observe that the context descriptions of lower level patterns are shorter because they refer to other patterns. For example, the context of REHEARSAL:

> You are planning an ONLINE TRAINING with several participants and you are currently preparing the structure. The training content and TRAINING OUTLINE are fixed in principle.

The context statement is brief but it refers to ONLINE TRAINING and TRAINING OUTLINES. The whole situation defined by these two patterns is part of REHEARSAL's context as well.

The context of CONTROL MONITOR is more detailed:

> In an ONLINE TRAINING the trainer will demonstrate the use of software on a screen that is broadcasted to participants who are located at another site. In opposition to classroom settings he will not see exactly the same image on the screen. Participants will see the screen in their browser window with a delay of time, lower screen resolution and a lower refresh rate. Furthermore, the broadcasted screen is usually integrated into a conference system which brings additional panels on the screen (e.g. video-image of the trainer, chat window).

The whole context describes a specific aspect found in ONLINE TRAININGS. This special context occurs in all online trainings. The potential need of a CONTROL MONITOR is therefore part of the solution of ONLINE TRAININGS:

> To check how demonstrated work steps are broadcasted to the participants, a second monitor can work as a CONTROL MONITOR. This monitor shows the broadcasted screen from the perspective of the participants.

The solution of ONLINE TRAINING refers to the pattern CONTROL MONITOR. Thus, it saves to describe the details of the situation because they are described in the context section of CONTROL MONITOR. Likewise, the solution of ONLINE TRAINING refers to CO-HOST, PAUSES FOR QUESTIONS, and PREPARED EXAMPLES. The solution builds on those patterns and the solution description brings the effects of the patterns together by relating them into a whole structure.

4.4.3.5 Follow-up problem solving by other patterns

The application of a pattern has side effects; it brings us into a new situation. Once we have chosen a specific path or solution, we will encounter problems that are particular to that decision. To tackle these follow-up problems, each pattern has an obstacle section. For example, two obstacles for ONLINE TRAINING are:

> ***Preparation Gap***. There is a huge difference between planning and running a training. Only if you run and implement the training, the real challenges in operating the software and applying meaningful actions will reveal. Whether the instructional outline works can only be tested if you run the training. Therefore, do a REHEARSAL before starting a training with real participants.
>
> ***Technical Problems.*** According to Murphy's law, if anything can go wrong, it will! If anything goes wrong during the live event this could cause inconvenient delays and may let the participants quit even before the event started. Furthermore, technical errors look very unprofessional – even if the trainer or host cannot be accounted for the error. Therefore, always check the technology in advance, fix any broken components, and analyze sources of failure.

Obstacles describe problems that occur in the context of choosing one solution. For both obstacles found in the context of running an ONLINE TRAINING there are further solutions offered. The preparation gap can be resolved by running a REHEARSAL. In fact, the first paragraph discusses the part of the resulting context that requires a REHEARSAL. Because the resulting context is discussed in detail, the context description of REHEARSAL can be very brief; the situation for REHEARSAL is described as an obstacle of ONLINE TRAINING. That a REHEARSAL solves the problem of the preparation gap is also indicated by the pattern-typical "Therefore" that links a solution to a discussed problem.

4.4.3.6 Patterns within patterns

The second obstacle, Technical Problems, is not linked to another pattern. Instead the solution is given in a single sentence: "Therefore, always check the technology in advance, fix any broken components, and analyze sources of failure." This advice is indeed a pattern and one could easily extend the pattern language with a pattern "Technology Check".

Blowing up small solutions into patterns, however, could lead to an explosion of pattern descriptions. The pattern DO NOT DISTURB is an example for such a blown up pattern. Instead of writing a whole pattern that in its main parts only states the obvious one could have integrated the problem, forces, and solution as an obstacle. Discussing the benefits of liabilities of a DO NOT DISTURB sign certainly is not innovative and the text sounds constructed:

> **Benefits**
> - Simple and effective, easy to implement
> - Everybody knows what is going on
> - Running a training without disturbance
>
> **Liabilities**
> - People cannot contact you in an emergency
> - Explicit "Do not disturb!" hints can provoke and even invite trouble makers
> - People might think you are taking things too seriously

The motivation to write this pattern was twofold. At the one hand, the obvious is often forgotten. Disturbance during trainings is a serious problem and to remind people of the sources of distraction is important:

> **Problem**
> Extraneous disturbances such as ringing phones, chatting colleagues, students or any operating noises distract the trainer, interfere the quality of audio broadcast and irritate the training flow.

The second reason to write the pattern was to show that everyday designs are meaningful and that there is a good reason behind the artefacts we take for granted. The discussion of forces is an exercise to explicate the reasons for a DO NOT DISTURB sign:

> **Forces**
> ***Usual Fuss*** Colleagues, guests, or students cannot know that an online training is broadcasted live from your office and that entering the office or knocking at the door will disturb. In particular if you usually have a policy of open doors and people are used to just entering other people's offices, you have to communicate that there is a "special situation".
>
> ***Interruption*** To turn off annoying operating noises or to get rid of someone during the training costs time, destabilizes the Zen of a training session and can cause embarrassing situations in the virtual meeting room.
>
> ***Absence*** To leave the desktop temporarily is not an option because a virtual training room without a trainer is very irritating.

All this said, in a revision of the pattern language it is recommendable to include the pattern as an obstacle in the description of ONLINE TRAININGS. But it demonstrates an important lesson. A solution consists of sub-solutions. Each aspect of a solution can be described directly in a solution section or one can use sub-patterns to build the solution. Both options are valid and sometimes it is wiser to not extract sub patterns if their granularity is too small.

Likewise the follow-up problems - the problems that arise in the context of applying a certain pattern - can be handled in the pattern itself or in sub-patterns.

4.4.3.7 Interplay of patterns

The solutions of the obstacles contribute to the whole solution in the same ways as the other sub-patterns in the solution section: CO-HOST, PREPARED EXAMPLES, PAUSES FOR QUESTIONS, and CONTROL MONITOR. The obstacles are just highlighted because one has to be particularly careful at that part of the solution. If we remember the path metaphor, each of the sub sections of a path contributes to the whole path. But some sub sections, for example an area with thorny cactuses, need more consideration.

The solution balances the forces. The benefits are very well balanced forces, the liabilities are forces that are not yet balanced – one still has to find particular solutions – and the obstacles are solution parts that are particularly hard to balance. The balancing is so hard that for some obstacles it is discussed in detail in a separate pattern.

Patterns do not only affect other patterns of a higher level. The application of two patterns at the same time affects both patterns reciprocal. For example, one of the benefits of CONTROL MONITOR relies on another pattern:

> **Benefits**
> - By looking at the second monitor, one changes his audio comments. Rather than saying "Now you see," one says "In a moment you will see". Hence, the audio comments are more appropriate to what actually happens on the screen.
> - A second monitor enables the trainer to observe chat messages and read them in planned PAUSES FOR QUESTIONS.

The first benefit makes the online training directly better. The second benefit can only be realized if there are also pauses to read the chat messages. The benefit relies on both patterns.

4.4.4 Conclusions

A written representation is not only a way to communicate about the patterns we have in our heads. The written representation (both structural visualizations and text) is also a cognitive tool to "pull out" our implicit knowledge about structures in the world. This process can help to understand what we intuitively know – or think to know. Once we start to reason about our implicit knowledge structure we may encounter incoherent ideas.

4.4.4.1 Implications of pattern description formats

The description format defines which information will be explicated. Each pattern and each of its actual implementations has always more implicated information than captured. However, the description format directs to different questions and answers. For example, if a descpription format includes an "Intent" field it is ensured that the context description does not only describe the general situation of applicability. It also states the goal that needs to be focussed on in order to make this pattern attractive. The wording of the pattern fields also impacts the focus of analysis. The "Use when" and "What else can I do?" fields for patterns of interactive graphics have led to an enumeration of examples whereas the "Context" field for the patterns of online trainings is more flexible. The desciptions of situations are expressed in more general terms and what is covered changes from pattern to pattern.

Each description field asks a question and requires an answer from the pattern authors. Therefore, these fields are a great tool to reflect upon known solutions. The drawback is that there might be fields where the answer is unknown. A solution might work as a whole and can be justified reasonably without knowing all answers. For example, if a designer has used a solution frequently in specific situations, the context field can easily be described. However, the designer may not be aware in which other situations the pattern works as well or fails. In this case, a description field "Contraindications" might trigger unjustified speculations.

A pattern description is a set of statements and each statement can be falsified. Reseachers can falsifiy specific examples of the context, specific parts of the solution, assumed consequences, the expected interplay of forces and the relations between context, problem, forces, and solutions. A pattern description could contain some statements that are false but the pattern itself could still be justified. For example, DYNAMIC LABELS is a valid pattern but its solution description (see section 4.4.1.9) contained inappropriate suggestions for the implementation. Hence, the pattern is correct whereas the description is not entirely valid.

4.4.4.2 Implications of pattern relations

The interplay of patterns is the real benefit of pattern languages. While relations between the patterns of interactive graphics only point out their categorical relations, the relation in a pattern language point out how the patterns affect each other. The quality of an ONLINE TRAINING is affected by the sub patterns such as REHEARSAL or TRAINING OUTLINE. In contrast, the quality of an ON / OFF BUTTON is not affected by objects that move in synchronization (SYNCHRONIZE OBJECT MOTION). At least such dependencies are not discussed in the pattern descriptions.

The descriptions of single patterns in pattern languages can be briefer because the descriptions can refer to other patterns instead of elaborating all details. By referring to other solutions it is also more convenient to name alternative implementations and express the solution in more abstract terms. The abstraction can still be unfolded because the generation process is described in other patterns.

4.4.4.3 Relations of interactive graphics vs. relations of online training patterns

The connections between the patterns of interactive graphics define specializations and alternatives of patterns. The hierarchy is defined by IS-A connections. For example, the Transport Object pattern is a special case of SYNCHRONIZE OBJECT MOTION. The ROLLOVER BUTTON is an alternative to ON / OFF BUTTONS; both buttons are specializations of a more general category of interactive buttons. The patterns SANDBOX and NO-GO-AREAS are inverse in that one specifies where to go (stay in the area of the sandbox), and the other where to not go (stay out of the no-go-area).

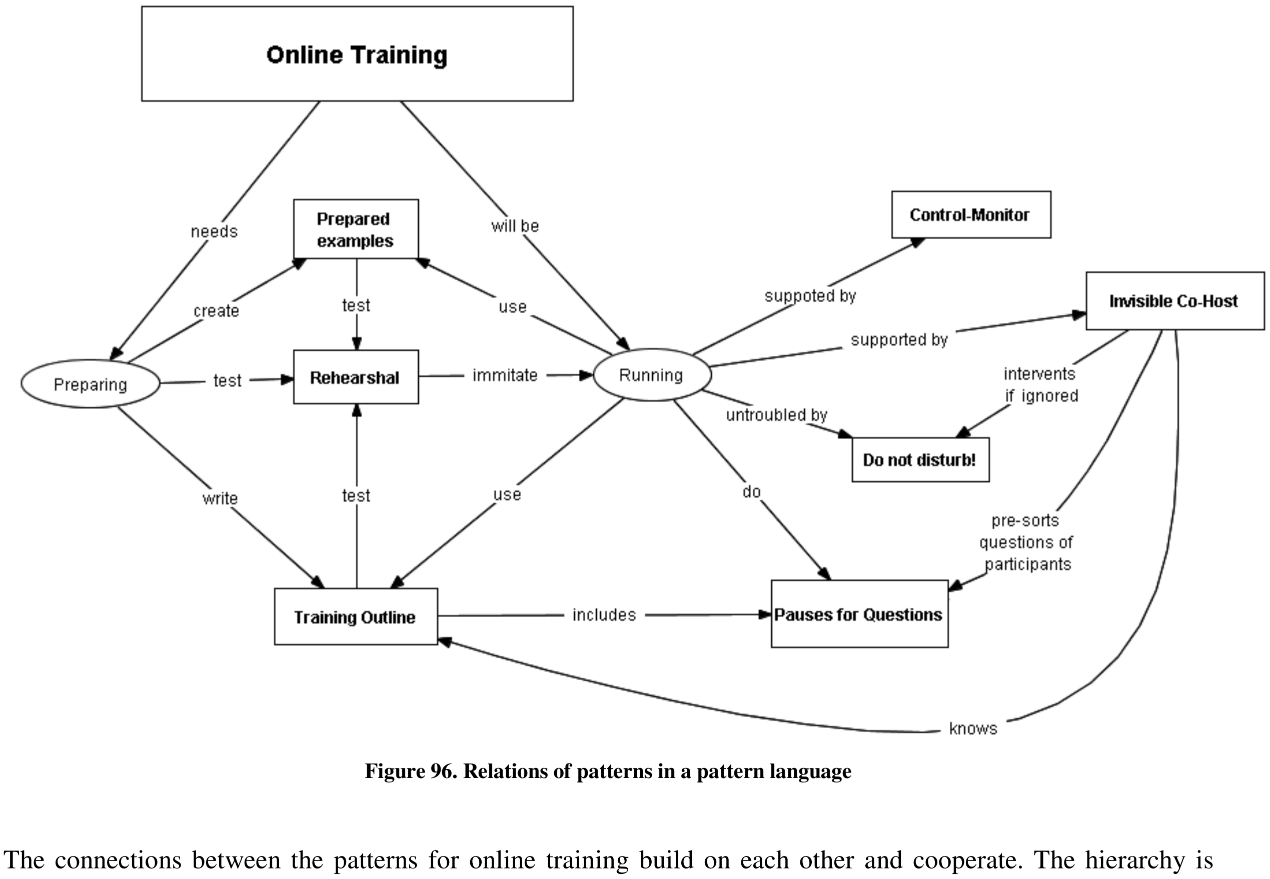


Figure 96. Relations of patterns in a pattern language

The connections between the patterns for online training build on each other and cooperate. The hierarchy is defined by HAS-A and USES-A connections. For example, a good ONLINE TRAINING needs a TRAINING OUTLINE to be prepared. PREPARED EXAMPLES are meaningful if they are used in an ONLINE TRAINING and combined with other content. The TRAINING OUTLINE shapes the required examples, and the examples shape the TRAINING OUTLINE. The patterns affect each other. One pattern needs the other patterns to generate a greater whole. That is why the patterns and their connections form a pattern language.

Some of the patterns overlap with other languages. For example, PREPARED EXAMPLES are not only meaningful for ONLINE TRAININGS. By integrating PREPARED EXAMPLES in the pattern language of ONLINE TRAININGS, however, the whole contexts of the language affects the form of PREPARED EXAMPLES. The prepared examples unfold into a specific direction because of the more specific context.

While all of the patterns of the pattern language affect each other, the end product, an ONLINE TRAINING, also has relations to other e-learning patterns, such as E-ASSESSMENT, MULTIPLE CHOICE, TESTS, SIMULATIONS etc. These patterns may play together to build an e-learning scenario. However, they are more independent from each other. The relations between different e-learning patterns are of the same type of relations between patterns of interactive graphic. For example, an ONLINE TRAINING is an alternative to an ON-SITE TRAINING. Both are sub categories of the more general pattern TRAINING. These relations between e-learning patterns, however, do not define a language.

Figure 97 shows the result of a brainstorming about potential e-learning patterns. Patterns with similar purpose have been clustered within the orange category boxes and arrows indicate the relations between each of the categories. The graphic shows that ONLINE TRAINING is treated as one pattern but we have seen that we can formulate a whole language to build this one pattern. Likeweise, INTERACTIVE GRAPHICS are considered as one pattern but we have seen that there are many different patterns of interactive graphics. In both cases these are entry patterns that describe the form on a high level. ONLINE TRAINING can be compound of other patterns. INTERACTIVE GRAPHICS can be differentiated into many other patterns.

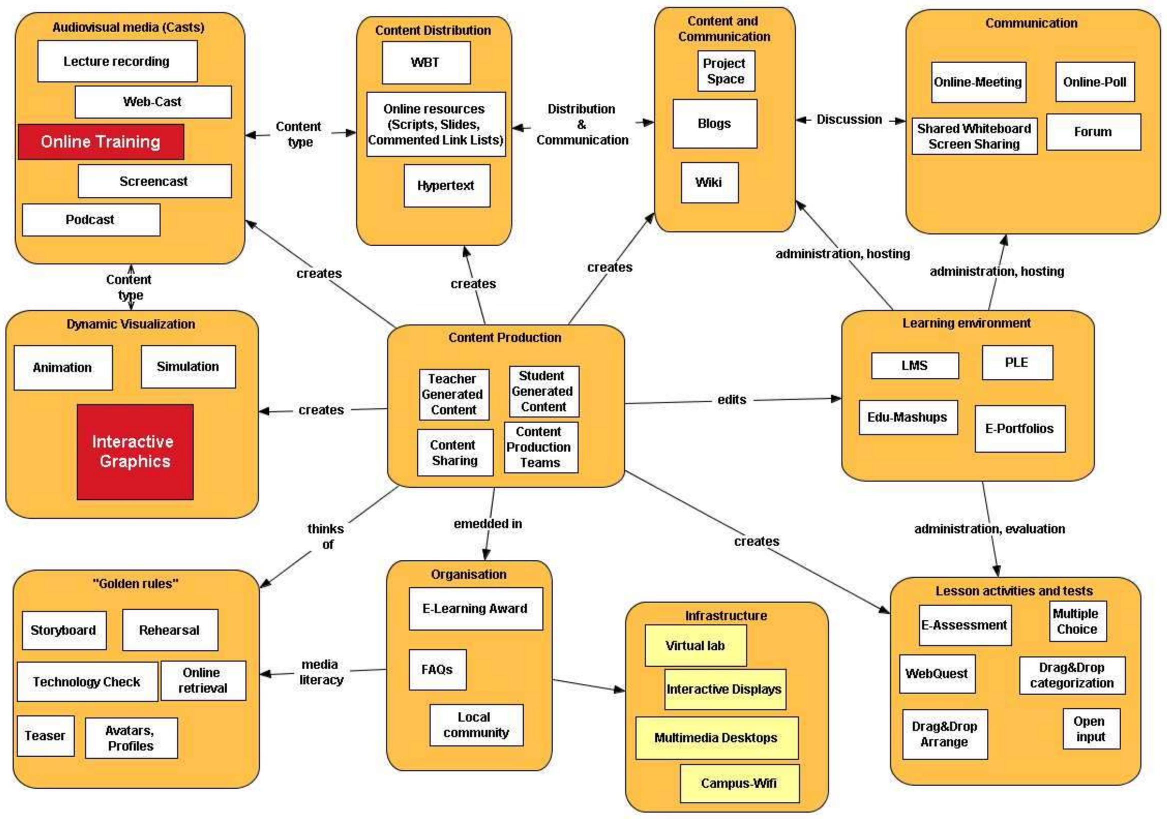


Figure 97. Relations between e-learning pattern candidates. The patterns relate to each other but they do not build on each other.

That the collection of patterns for interactive graphics and the collection of e-learning patterns are not yet part of a proper language does not mean that they never could. For example, the patterns of interactive graphic could play together to create an interactive application. Likewise, patterns for e-learning could play together to build an e-learning course. Good examples for pattern that are grown into a language are the patterns to design groupware systems (Schümmer and Lukosch, 2007), patterns for fault tolerant software (Hanmer, 2007) and patterns for fearless change (Rising & Manns, 2005).

4.4.4.4 Related patterns and the limits of languages

However, pattern languages of a too broad scope have their own problems. For example, an ONLINE TRAINING, a PRESENTATION, a SEMINAR, or a SIMULATION could all benefit from PREPARED EXAMPLES. However, a PREPARED EXAMPLES pattern that takes into account the four different contexts needs to me more general. While the PREPARED EXAMPLES in a pattern language for ONLINE TRAININGS implicitly were PREPARED EXAMPLES FOR ONLINE TRAININGS, the more general pattern would be "PREPARED EXAMPLES FOR ONLINE TRAINING, PRESENTATION, SEMINAR OR ONLINE TRAININGS" - or "PREPARED EXAMPLES FOR E-LEARNING SITUATIONS". However, by making PREPARED EXAMPLES more general, it looses some of its specific gestalt; it becomes less whole because it takes less of the contextual specifics into account. In order to serve in very different contexts its form has unfolded less, leaving many decisions open. We cannot use the same PREPARED EXAMPLES pattern when it is meant to be used in broader contexts than ONLINE TRAININGS.

Depending on the scope of the design problem, sometimes systems of patterns are better than pattern languages. A pattern language to build a thing must always contribute to what that thing is. Alexander (1979) points out that each project needs its own pattern language. If we have a very general category, such as e-learning, we cannot provide a general pattern language to build that "thing" of e-learning. The reason is that there are many different forms of e-learning and each form would require a different configuration of that language. However, if we have a more specific thing to build, such as an Online Training, we can provide a pattern language that takes the specifics of that form into account.

4.5 Refinement of the framework

Based on our empirical evidence we can refine our pattern framework. We have discussed three different „places“ of patterns: patterns in the world, patterns in our heads, and explicate pattern descriptions. We had argued that patterns of the world are constructed as schemas in our heads. These schemas are implicit structural representation just as the information of a structure is implicit given in the structure. The pattern descriptions are a way to explicate these ideas - theories and views on solutions.

In our original version of the framework we only considered an inductive one-way process: observable recurring structures in our world lead to patterns in our heads (pattern acquisition) and these mental structures (schemas) can be explicated as pattern descriptions (pattern writing). Based on the documented patterns, other individuals can start to re-use the solutions and create new instances of the pattern (pattern application)

However, the theoretical foundations for the different pattern types and their empirical testing suggest more relations for the framework. There are feedback loops between the patterns in the world, in our heads and the pattern descriptions.

4.5.1 Patterns in our head shape the world

First of all, the structures that we observe in the world are shaping our schemas. But it is not only the world that makes up our reality. Our schemas are influencing the world and how we see the world in many ways. If we have a good solution in mind, based on previous experience, we are likely to implement that solution again. The implementation, however, is a new instance of that pattern. This changes the world structure. Particularly in the field of software design the models and metaphors an engineer works with lead to new digital artefacts and new work or business models. In a way, we are creating more evidence for the pattern – whether that is good or bad.

The authoring tool for interactive graphics is an example how a specific way of categorizing existing graphics into patterns leads to more instances of them. The tool offers a convenient way to create exactly these forms. Likewise, many standard software systems have impact on the artefacts created by their users.

Another example is how the style of websites has changed over time. Web designers copy the styles of other designers – consciously or not – because they constructed a schema of that style. Creating another site in the same style makes it more likely that other designers will pickup the style because it recurs more frequently. The higher frequency does not only increase the chances that the style or structure is seen at all but is seen several times – the ground for schema equilibration. Schemas also govern our decisions and interventions in our daily lives, our values and behaviours that will have effect on the world – how we treat the environment or how we communicate with each other. And schemas influence our views of the world. The individual theories or ideas we have about the world influence what we see of the world – the perceived world structure. Thus, which patterns we find and which data we can get from the world depend on our theories – the ways of looking at the world.

Thus, the individual ideas and implicit views on the word influence which patterns we perceive and which ones will be instantiated again and again.

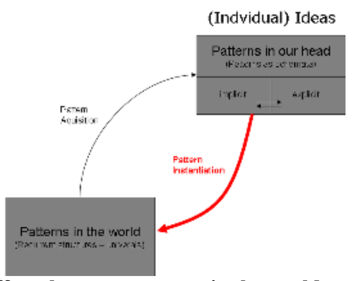


Figure 98. Reciprocal effects between patterns in the world and patterns in our heads.

4.5.2 Putting ideas in our minds

The explication of ideas is a way of sharing our insights and views. A pattern description explicates the solution form the authors have in their minds because they have correctly or incorrectly observed it in the world. Other people who read the explication can judge whether it is coherent with their world. Again, the writing process is not a one way road. Writing down ideas of patterns changes those ideas. Moreover, the ideas can be picked up by others. By reading about a specific structure, we are guided to construct our own schema about applying the pattern.

When we read a pattern document we only get an explicated description. In order to implement it we have to build our own implicit representation – a pattern language becomes only alive in our heads (Alexander, 1979). The implicate schema makes the structure whole because it integrates all of the structural relations as *one*. Only if we have an idea of what we want to build we can actually build it. Ideas, however, are the implicate representation not the explicate description of them. Pattern descriptions can help to get the right ideas, i.e. ideas that represent solutions that have worked in the past – not just ideas out of the blue.

By putting these ideas in the heads of designers, their design process may change: they start to implement new instances of the pattern. The Gang of Four patterns are now ubiquitous. They have been real when they were documented but now they are even "more" real because there are millions of implementations – not because these solutions are "natural" but because people have read about them and applied them. There are more *realizations* of them because of the written descriptions. A documentation of patterns, hence, can change the world. Linda Rising and Karl Rehmer (2009) have started to write wonderful patterns for sustainable development. That patterns can change the world by changing our views is a promising hope.

Thus, patterns in our heads do not only lead to pattern descriptions. The pattern descriptions also lead to new patterns in the heads of other individuals and may lead indirectly to more pattern instances.

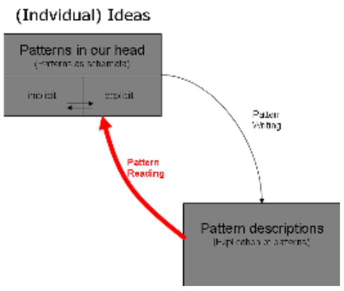


Figure 99. Reciprocal effects between patterns in our head and pattern descriptions.

4.5.3 Pattern mining

Whether a pattern description captures a structure that can actually be found in the world is another question. We can easily describe a pure fictional pattern that does not exist at all. Moreover, if the authors are not experienced in their domain they might have constructed the wrong patterns. But even if we have adequate patterns in our head, the explicate description might be incomplete or contain wrong elements as we have seen in section 4.4.5.2 (the solution provided a misleading advice how to implement the pattern in PowerPoint).

Therefore, we need ways to ensure that the pattern descriptions actually represent meaningful patterns of the world. We need another relation in our framework to show the process of finding pattern descriptions that are tested and justified. A written pattern should be the result of thoughtful pattern mining, a process which extracts "nuggets of wisdom". We can say "wisdom" because their insights are grounded in many reviews of actual designs. Pulling out this knowledge is like mining (Rising, 1998) for nuggets; the core of the pattern is pointed out; the noise of actual instances is taken away. This mining process is a process of cognition – just as any theory building. A pattern author often reconsiders the artefacts and examples her or his pattern is based on. Writing down the pattern, too, is an active process that tries to resemble the universal structure of the pattern. Pattern descriptions are proposals to specific views on the world, and on design problems in particular. However, their validity needs to be tested as any theory.

Thus, we need to add a relation to show that patterns in the world (reality) are mined as pattern descriptions (theories about that reality).

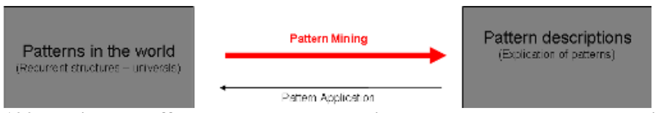


Figure 100. Reciprocal effects between patterns in the world and pattern descriptions.

4.5.4 Refined framework

If we integrate the new identified relations, we can refine our framework. We consider the *patterns in our heads* as individual ideas about realities. These ideas can be learned directly by observing the world (*pattern acquisition)* or indirectly by reading about existing solutions (*pattern reading*). *(Individual) ideas* can be explicated by *pattern writing* which results in *pattern descriptions*. As the pattern description is a proposed way of looking at the world we consider it as a theory about the actual patterns in design structures. Such a theoretical description can be shared with others who can use it for their own design process. Based on individual ideas about designs (either directly aquired or learned from reading) new pattern instances will be created (*pattern instantiation*). The creation of new and original artefacts will in itself lead to an evolution of reality (e.g., the invention of personal computers created a new reality).

To get the right views (*theories*) on patterns is very hard. There are many ways of seeing the world and organize its structure; there is always doubt whether we have seen enough of the world (*reality*) to identify sufficiently stable patterns. Patterns that have been identified in a *pattern mining* process are fallible in principle and can be falsified empirically. If a pattern constantly fails, it needs to be rejected – the pattern itself, the pattern description or both. Successful *pattern applications*, on the other hand, are corroboration (not proofs) for the adequateness (not truth) of the insight provided in a pattern description.

Both *pattern mining* (finding theories that adequately capture recurrent good designs) and *pattern application* (creating new good designs based on these theories) can only be achieved indirecty in a social process of individuals.

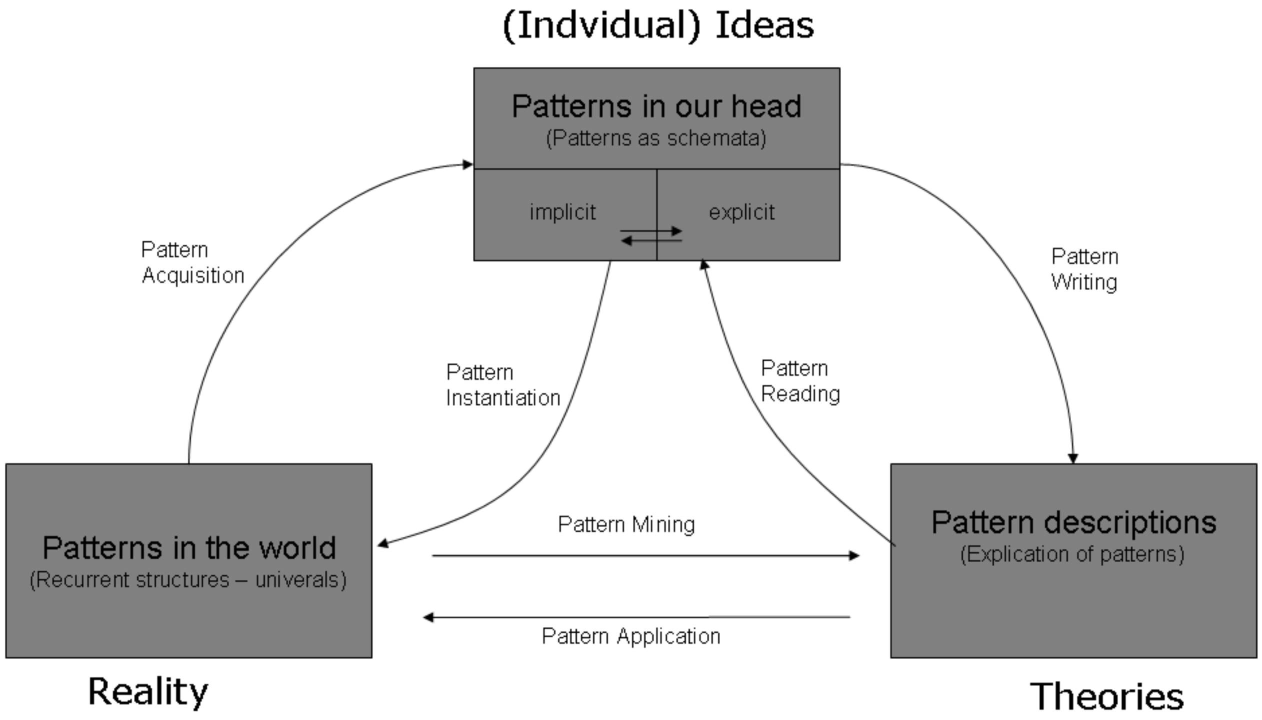


Figure 101. Refined framework

5 Conclusions

The refined framework shows that pattern descriptions do not capture absolute truths. Although a pattern description is the result of a pattern mining process this is never achieved directly, automatically or formally. We cannot automatically analyze program code for recurrent patterns because the code is not the program's whole world structure. That structure includes the program's meaning – its intent – as well. Very often pattern writing is a social process. A team of authors shares their views. A shepherd and the participants in a Writers' Workshop comment on the pattern and express their views. Collaborative pattern descriptions can be created using a Wiki. Cress & Kimmerle (2008) argue that there is a co-evolution of the structure described on a Wiki page and the cognitive structure of its users: A reader of Wiki pages adapts his cognitive structure if he discovers new insights (accommodation). If the pattern description reflects his own ideas, his schemas will be strengthened (assimilation). Furthermore, he can change the pattern descriptions if it does not fit to his knowledge (accommodation of Wiki page) or add more examples for the pattern (assimilation). Hence, Wikis are adequate tools to assimilate and integrate multiple views on patterns (Kohls & Wedekind, 2008); if individual ideas are different, Wikis are tools to merge them collaboratively. It is no coincidence that Wikis are suitable for pattern descriptions since the first Wiki was designed for a pattern repository.

Since schemas are constructed there may be different views on the world and from the same design artefacts different patterns can be derived. The sections about forms and information (3.2.3 – 3.2.8) have shown that in different perceived world structures (i.e. different parts of the real world such as different domains, different expertise, different cultures etc.) we will find different patterns. These patterns are real but not all of them are equally meaningful. The adequateness of identified patterns changes if the perceived world structure is extended (i.e. we get more experience or visit other cultures). Moreover, the perception of a structure is never perfect – our senses and apparatus are not precise enough to exactly grasp the whole underlying structure. Besides this imperfection we are also guided by what we are focussing on. Depending on what we have seen of the world, we will create different individual schemas. The structure represented by an equilibrated schema is isomorphic to the structures of the real world that it captures. For example, an individual schema is isomorphic to the universal structure of cars. However, the structures are not exactly the same. They are only symmetrical, the universal form of cars is symmetrical to my idea of cars and most people's ideas of cars – but they are not the same. They are also stored differently: The universal of cars is *in* the forms of all cars; new cars change that universal. The schema that I have about cars is represented by a pattern in my head – a macro-state or a high level of patterns of neuronal connections. Both the universal and our differently constructed schemas of cars are implicit representations. A description of a car is an explication of (our hypotheses about) essential features and relations, a representation of a car (a diagram or a model) is an explicate expression that resembles the implicit relations.

Likewise, the symmetry that is found between all instances of the BRIDGE pattern is covered in descriptions of the BRIDGE pattern. The descriptions mean the same universal thing but they are not the universal itself. A pattern description provides insights by explicating a way of seeing at a structure (such as software architectures or educational scenarios). It might not be the only way of seeing at the structure but it should be one that is meaningful and coherent with the actual structure.

Now that we have clarified the different types of patterns and their relations, we need to ask whether the framework and our discussion about patterns help to address the challenges we had identified in the introduction. The next sections will revisit the research questions, suggest answers, draw conclusions and point to further research questions. Figure 102 provides an overview which chapters can provide answers to the research questions.

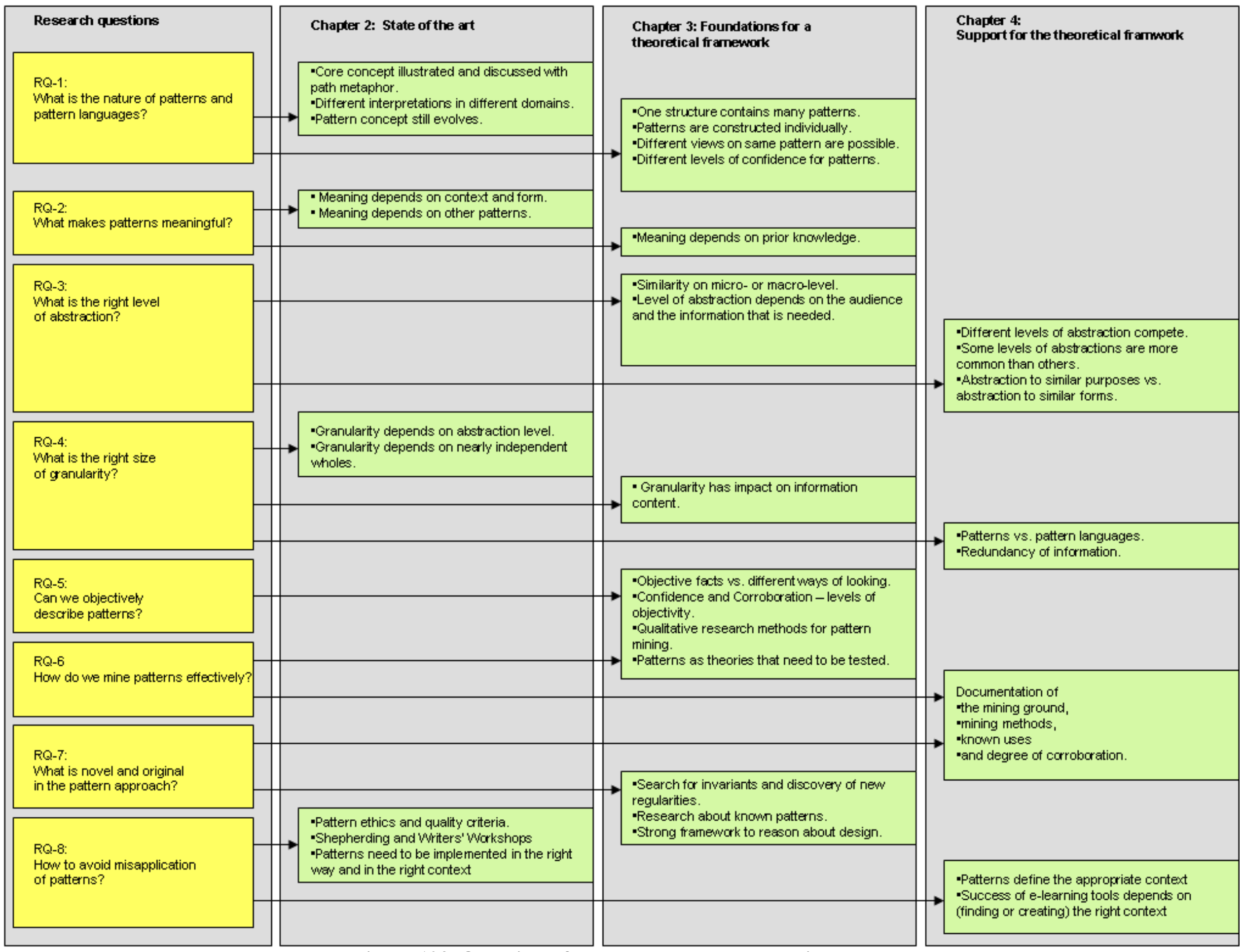

Research questions
RQ-1: What is the nature of patterns and pattern languages?
RQ-2: What makes patterns meaningful?
RQ-3: What is the right level of abstraction?
RQ-4: What is the right size of granularity?
RQ-5: Can we objectively describe patterns?
RQ-6 How do we mine patterns effectively?
RQ-7: What is novel and original in the pattern approach?
RQ-8: How to avoid misapplication of patterns?
Chapter 2: State of the art
•Core concept illustrated and discussed with path metaphor. •Different interpretations in different domains. •Pattern concept still evolves.
• Meaning depends on context and form. • Meaning depends on other patterns.
•Granularity depends on abstraction level. •Granularity depends on nearly independent wholes.
•Pattern ethics and quality criteria. •Shepherding and Writers' Workshops •Patterns need to be implemented in the right way and in the right context
Chapter 3: Foundations for a theoretical framework
•One structure contains many patterns. •Patterns are constructed individually. •Different views on same pattern are possible. •Different levels of confidence for patterns.
•Meaning depends on prior knowledge.
•Similarity on micro- or macro-level. •Level of abstraction depends on the audience and the information that is needed.
• Granularity has impact on information content.
•Objective facts vs. different ways of looking. •Confidence and Corroboration – levels of objectivity. •Qualitative research methods for pattern mining. •Patterns as theories that need to be tested.
•Search for invariants and discovery of new regularities. •Research about known patterns. •Strong framework to reason about design.
Chapter 4: Support for the theoretical framwork
•Different levels of abstraction compete. •Some levels of abstractions are more common than others. •Abstraction to similar purposes vs. abstraction to similar forms.
•Patterns vs. pattern languages. •Redundancy of information.
Documentation of •the mining ground, •mining methods, •known uses •and degree of corroboration.
•Patterns define the appropriate context •Success of e-learning tools depends on (finding or creating) the right context

Figure 102. Overview of answers to research questions.

5.1 *Multiple meanings of "pattern"*

Research question RQ-1: What is the nature of patterns and pattern languages?

5.1.1 Domain influences the pattern concept

From all our discussion we have learned that the same form - the same structural relation - has different meanings in different contexts. Likewise the same concept – the invariant of the pattern approach – has different meanings to different domains and different peoples. This diversity is valued (Coplien, 2004). The pattern concept has many implications; the implicit structure is unfolded different in different contexts. Chapter 2 on the structure of patterns has tried to seek some invariants that are commonly agreed on. But we have also shown the different meanings, interpretations and uses in diverse fields. We have also pointed out some misunderstandings. For example the concept of a pattern language implies a network of patterns; however, a network of patterns does not automatically make a pattern language. Rather, we have followed Noble and Biddle's view (2002) and considered patterns from a semiotical perspective. This view explains different names for the same pattern descriptions and different pattern descriptions for the same pattern.

5.1.2 Different ways to construct patterns

Though we have argued that patterns are universals (patterns in the world), the same structure can be divided (real-ized) in many different ways. Thus, patterns are nested, overlapping, and found on different levels of abstraction. Since the experienced structures of the world are different for each individual we have seen that the patterns in our minds – the schemas – are constructed individually and depend on culture and prior knowledge. Hence we can have different views on the same pattern or find different patterns altogether. When we share the same experiences, however, it is likely that in the process of accommodating our mental structures become more and more coherent with the actual (perceivable) structures and the ideas of other persons. Pattern descriptions are an attempt to capture this invariant. Written patterns are explications of implicit structures (the real structures in the world, and the structures in our minds) that can be more or less appropriate. They can also be plainly wrong if the proposed structure cannot be found in the world at all.

5.1.3 The pattern concept still evolves

While chapter 2 has discussed mostly views that are agreed on by many people, it has also shown that there are questions related to the pattern approach that are not fully understood or accepted. In particular, Alexander's later work has not been appreciated very much. While many ideas of *The Nature of Order* are present in *A Timeless Way of Building*, the focus on Alexander's fifteen fundamental properties of life and the structure preserving transformations to achieve them are a newly taken perspective. The fifteen properties have been discussed rarely in the pattern community. At the one hand, the properties need to be interpreted for the various domains such as software design, human computer interaction, or pedagogy. Moreover, the four volumes of *Nature of Order* are not very accessible. Jennifer Quillien's (2008) discussion in *The Delight's Muse* is a very good starting point to understand Alexander's new ideas. The work has been discussed lately on PLoP conferences and it is possible that in the future the fifteen properties become more important in discussing good and beautiful design. We also discussed the properties with Jennifer Qullien at a workshop on educational patterns in Vienna (November 2012).

5.1.4 Future work

To test whether the fifteen properties do relate to good design one future research project could be to investigate which of the properties are present in patterns that are widely agreed on. This question is also interesting for the patterns of interactive graphics and the pattern language of online trainings. Another research question relates to the clustering and hierarchical relations of the fifteen properties – how do the properties relate and influence each other?

The path metaphor has worked well to illustrate many of the core concepts of patterns. While the detailed discussion provides background information it is sometimes confusing to individuals who are not familiar to the pattern concept. The author is working on a revised version for beginners that is planned to be part of the introduction package for the *PLoP conferences.

5.2 Meaningful patterns

Research Question RQ-2: The challenge is: what makes patterns meaningful?

5.2.1 Meaning depends on context and form

Meaning unfolds only in contexts as we have seen in chapter 2. Any tool, method or knowledge can be meaningful in one environment but have no meaning in a different context. Meaning is the interplay of context and form – their fitness. Fitness of the two is meaning. Things that have no meaning in their current context will hardly survive (Mitchel, 2009).

The meaning of a pattern, hence, depends on the environment. Parts of that environment are the individual agents who use, create, or apply a pattern. The same pattern, therefore, can be meaningful to one person and meaningless to another. Reasons can be different interests, domains, or different problems one faces.

5.2.2 Meaning depends on prior knowledge

Only if an individual has enough prior knowledge to understand the pattern it becomes meaningful to him. For example, if audience A knows what a "Brainstorming" is we do not need to inform the audience about the structure of a BRAINSTORMING. The word "Brainstorming" denotes the form category and if we have this category in our mind we can unfold it without further explanation. Likewise, in science the formula $E=mc^2$ is only meaningful if one understands the symbols. The implicit meaning of the formula takes volumes of books to be explained (explicating all the underlying physical assumptions).

At the last *PLoP conferences there have been several approaches to teach and learn patterns by pattern briefs (very short and abstract pattern forms). Deborah and Ilja Preuss have developed a card game to learn using the *Fearless Change* patterns from Rising & Manns (2005)[34]. Takashi Iba (2012) and his colleagues use very brief and abstract pattern descriptions that stimulate discussions and the sharing of experience about patterns. In these

[34] Claudius Link presented the game in an open space session.

cases the full meaning is not provided in the description of the patterns briefs but constructed in personal communication. It requires prior knowledge from the participants to unfold the pattern.

5.2.3 Meaning depends on other patterns

To understand a given structure we rely on schemas that we have constructed already. We need to know what IMPULSE QUESTIONS, STICKY NOTES ON PIN BOARDS and POSTPONING JUDGMENT mean if we want to refer to them in a description of a BRAINSTORMING. Likewise, if we want to integrate a BRAINSTORMING into another pattern, such as SIX THINKING HATS, we assume that readers have knowledge about brainstormings - either from their own experience or by reading another pattern. This is also supported by Alexander's view on pattern languages: the richer the language in our heads, the richer the designs and solutions we can achieve. Pattern descriptions likewise depend on prior knowledge of readers. If the knowledge is already rich, the description of a new configuration can be very brief. We do not have to describe what an IMPULSE QUESTION is if we can assume that a reader knows the form category already. In this case, IMPULSE QUESTION means something to the reader. If we cannot make such an assumption we have to either include a description of what IMPULSE QUESTIONS are or refer to such a description, i.e. refer to another pattern.

5.2.4 Future work

The next language I am working on will cover creativity patterns. Patterns such as INVERSE VIEW, RANDOM IMPULSE, DREAM TEAM, or IDEA COMBINATION can work on both levels of detail: The written pattern discusses the context, problem, forces, solution details and consequences to understand how the creativity method works. At the same time a simple visual illustration with an abstract one-sentence-instruction can already stimulate thoughts. Likewise templates for whiteboards can support creative thoughts by putting the right questions on the table. The reason why briefer pattern solutions work in these cases is that the solutions or processes can be unfolded from common sense once the stimulus is given. Hence, the meaning of the patterns depends on prior knowledge and is rather socially constructed than fully captured in the pattern description itself. A future research question is how effective such formats are and which prerequisites need to be fulfilled.

5.3 The right level of abstraction

Research Question RQ-3: what is the right level of abstraction?

5.3.1 The right level of abstraction depends on the audience

There is not one right level of abstraction (Rising, 2007). However, there are wrong levels of abstraction. A pattern should neither be too abstract nor too specific. Too abstract patterns are not instructive because they can no longer describe an invariant gestalt. On the other hand, too specific patterns allow no adaption to the actual situation. The appropriate medium level of abstraction depends on the context and knowledge of the target audience.

Our study of interactive graphics has shown that there are different levels of abstractions that compete. A structure can be divided into different sub structures. Section 3.2 theoretically reasoned about this for structures in general; sections 4.3 and 4.4 showed the problem for finding the right patterns of interactive graphics. Our experiment revealed that there are levels of abstraction that are more common than others. From a technical point of view a ROLLOVER and ON/OFF BUTTON are very similar in implementation. However, they are used in different situations and have different consequences.

The right level of abstraction depends on the information that is needed. For example, for the technical problem of implementation these different contexts do not matter. For the pedagogical level the variations do matter.

We can conclude that the right level of abstraction is found when a pattern is informative for the target group, i.e. it provides enough information to create an actual instance of the pattern. This depends on prior knowledge of the target group and the structural variations in the form category.

5.3.2 Similarity on micro- or macro-levels

Similar structures are different microstates of a more universal macrostructure. For example, the various implementations of an OBSERVER are different microstates and the effect (or intent) of an OBSERVER is the macro-state. These microstates are all very similar – they have the same structural quality or gestalt. However, there are also macro-states that emerge from very different micro-states. A number of different ways (cross a bridge or take a boat) could lead to the same goal (cross a river) as we have seen in section 2.4.2.3. In this case, the emergent macro-state does not reveal anything about the actual form of the micro-state because there are alternative microstates for the same macro-state. In our experiment we found that some participants had classified the graphics by subject or pedagogical goal. At first glance it seems reasonable to focus on a pedagogical goal. However, a pedagogical goal (macro-state) most of the times can be reached by very different ways (micro-states). Naming a pedagogical goal is not very informative in respect to the actual steps that need to be taken. To describe the structure of steps – ways or methods – for a broad category that is based on similar effects (goals) rather than similar forms is very hard. One has to include all the variations to account for the different instances.

Common misunderstandings have their origin in considering only emergent features of a macro-level. When Plato was looking for the pure forms he also asked for the true form of justice. However, effects such as justice or happiness are emergent features. Very different forms (micro-level) can have the same emergent property (e.g. justice or happiness on a macro-level).

Patterns have to show similarities in their forms (micro-level) and in their effects (marco-level). However, considering only similarity of forms would be misleading as well. The same form has different meanings in different contexts. Therefore, a pattern needs to be on a level of abstraction where we find similarities in the context, the problem, the forces, the consequences and the solution form. That is we look for a level of abstraction that still has similarities in the whole.

5.3.3 The impact of abstraction

In retrospection, the research question about the right level of abstraction might have been the most important one because it affects the answers to almost all other research questions:

- To be meaningful (RQ-2) a pattern has to meet the appropriate level of abstraction in such a way that a target audience can re-construct the meaning and unfold new designs.
- There is a strong interplay between the abstraction, the scope and granularity of patterns (RQ-4). A specific level of abstraction suggests different scopes and granularities and vice versa (see 5.7.4)
- The objectivity of patterns (RQ-5) depends on how specific the pattern description is, i.e. how many concrete claims are stated in it.
- Likewise, in the process of pattern mining (RQ-6) the level of abstraction relates to the logical and empirical content of a pattern. "Do the right thing" from Joe Bergin is on the most abstract level, including everything that is right but telling nothing about what exactly that would be.
- Novel and original to the pattern approach (RQ-7) is that the abstraction level is required to preserve the gestalt – the whole of a design with its "Quality Without A Name".

5.3.4 Future work

The experimental findings are valid only for the sample of interactive graphics we used. It would be interesting to see how the abstraction changes if a different set of instances is used or more instances are added. While the sequence of presentation in the pair comparisons has been randomized there was no variation between the participants. Such a variation could show whether the sequence of perceived forms has influence on which schemas are constructed. The experimental environment could be reused to run such experiments in the future. Our basic research questions focussed on the acquisition of patterns. We did not test how stable the patterns are, i.e. would participants recognize the patterns on the same level of abstraction still after some weeks? A future research design could split or rerun the phases after defined time spans (e.g. repeat the classification tasks 3 or 6 months later).

5.4 Granularity of patterns

Research Question RQ-4: What is the right size of granularity?

5.4.1 Granularity depends on abstraction (and vice versa)

The granularity of patterns is closely related to the level of abstraction. On an abstract level we can describe patterns of larger granularity. If we want to describe an entire architecture or an educational scenario it is impossible to include all the details. A university that designs a curriculum cannot describe what a SEMINAR is each time. Hence, a curriculum description rather refers abstractly to concepts such as SEMINAR, LECTURE or

PROJECT WORK. A description of a seminar would be less abstract (providing more information) and have a smaller granularity. On deeper levels of a pattern hierarchy we find the most specific descriptions.

On the lowest level we find a whole particularization where everything is specified and concrete. Again, the path metaphor illustrates what this means. To describe a path to a goal we would not describe every foot step. Rather, the path is described abstractly by reference to milestones. How to get to the next milestone is a more concrete pattern of smaller granularity. On the lowest level a particular sequence of foot steps may detour around a stone at the left or right hand side. While we had described the path abstractly by milestones, an actual hike consists of all particular foot steps to the goal.

5.4.2 Granularity depends on nearly independent wholes

For the description of patterns we have to make decisions where we divide wholes into parts that can be handled. Our discussion of the path metaphor and the theoretical framework provides suggestions:

If the structural differences of various instances are low (micro-states are very similar) we can define a coherent pattern. Small variations are often discussed in the same pattern descriptions.

If there are very different ways of implementing a solution part, we need to capture the different gestalts as alternative solution patterns.

That very different ways lead to the same goal suggests that the same emergent property can be generated by each alternative. Each alternative is self-contained towards the emerging goal. Hence, we can split a whole into self-contained smaller wholes where different options serve the same function.

5.4.3 Granularity has impact on the information content

The thought experiment in 3.2 and the discussion of patterns and pattern languages in 4.4 has shown how the division of larger wholes into smaller wholes can impact the information content and the universality of patterns. Dividing a whole can reduce the complexity by extracting recurrent structure and redundancy. This is different to a simple “divide and conquer” which only distributes the complexity to manageable parts. However, splitting a large whole into its parts can also increase the complexity if there are many dependencies between the parts. A proper encapsulation is a core quality of good patterns.

5.4.4 Future work

The pattern language for online trainings consists of several patterns to successfully host online trainings. At the same time an ONLINE TRAINING can be a pattern of a larger pattern language or collection as we have seen in section 4.4.7, and particularly in figure 97. For each of the patterns of that large scale language there could be smaller languages to generate that pattern. At a workshop on educational patterns in Vienna (November 2012) we discussed to collect educational patterns. There could be patterns for various fields of education. The identified e-learning patterns could be part of a language for blended learning or the design of curriculums. The next steps for us are to map the existing educational patterns and identify gaps and research interests (further information can be found on eductionalpatterns.org).

In the past years the author has also collected patterns for the use of interactive whiteboards (Kohls, 2012a). While these patterns are categorized in a meaningful sequence they are not connected to a pattern language yet. Future work will interweave the patterns into a language based on the findings of this thesis.

5.5 Objectivity

Research Question RQ-5: Can we objectively describe patterns?

5.5.1 Objective facts

If we consider objective facts as facts that can be obtained from an object independently from the eye of the beholder, there are objective facts. More exactly we would say that different people get the same facts from the object – the facts are intra-subjective. However, which facts we get depends on how we look at the object. Theories are insights - different ways of looking at objects. Different theories may show different facts of an object.

5.5.2 Different ways of looking

One theory should provide coherent data. That means for one way of seeing we can get objective information. This information is implicit in the structure of that object and unfolded by the way of looking at it. But that does not mean that we have an objective way of measuring beauty or the degree of life as Alexander suggests. A simple counter-example is music. If we go to the charts of the iTunes store we will find the songs that many people like. Yet, there are many songs we dislike. Also, very often our own likes change. A song listened several times is more beautiful. A painting considered in a new light suddenly becomes more beautiful. Of course, the potential to be beauty is always in the object. Yet to unfold it, we need specific configurations of the contexts.

5.5.3 Aesthetical properties depend on context

It is likely that some objects are beautiful in many contexts, i.e. artefacts or events that are universally perceived as beauty without training. Other objects, however, depend on knowledge and training in order to see the beauty. They require more specific context structures to unfold their beautiful meaning. Therefore we object to Alexander's view that the degree of life is objective. In this question he contradicts himself because the meaning, beauty and degree of life of a thing can never be judged without context. In one of his experiments he shows a series of two pictures to participants and asks them which of the two pictures is more alive. The hypothesis was that people identify the same pictures to have a higher degree of life. However, Alexander required people to look at images at the right way (Alexander, 2002a). Looking at things in the right way actually does mean that the beauty of an object depends on the observer. Moreover, in his experiments Alexander never got 100% agreement. At a research workshop in Vienna (in 2009) we tried the experiment and got many disagreements. The experiment was once run at a VikingPLoP (personal communication with one of the participants) – it showed that participants disagreed on the degree of life they saw in the pictures. But even if there would have been a high agreement we could not infer that judgments about objects are always objective. That we like different music, different buildings and different places just shows that the quality depends on the context.

The objective qualities of an object are its potentials in different contexts. We can never capture all potential contexts. However, we can make statements about the effects of an object in common contexts.

5.5.4 Future work

While all of the patterns presented in this work have been discussed in Writers' Workshops there has been no large scale field study to the effectiveness of the described patterns. Schümmer and Lukosch (2007) have collected 37 patterns to design groupware systems. They have tested the patterns in over 150 student projects and the results have significantly improved (personal communication with Till Schümmer). The patterns of interactive graphics and online trainings have not been subject to such a test yet. Since patterns constantly evolve as the environment changes, the patterns also need to be re-evaluated for their appropriateness for today's technology. For example, the applicability of interactive graphic patterns needs to be tested for multitouch functionality and small tablet computers. Likewise new conference systems have emerged over the last years. Screencasts of small tutorials have become popular. Future research will have to answer how these new technologies and cultural trends affect the patterns for ONLINE TRAININGS.

5.6 Pattern mining

Research Question RQ-6: How do we mine patterns effectively?

5.6.1 Mining methods

We have learned about several ways how patterns are mined in practice. The methods and tools have been discussed in the section about the epistemology of patterns and some of these methods have been discussed in the empirical sections: expertise from our own projects, analysis of artefacts, observation of users, experimental setups (which is rather unusual in the pattern community), discussing the experience with peers and focus groups. Most of the pattern mining methods are variations of qualitative research methods. We have seen that the tools developed by the pattern community are particularly helpful to pull out the tacit knowledge of experts. The Shepherding Process and Writers' Workshops are not simply accepting or rejecting the findings of a paper. Of course the claims made in a pattern can be falsified. But Shepherding and Writers' Workshops go one step ahead because they test the accessibility of a pattern. The quality of writing matters as much as the right level of abstraction and granularity.

5.6.2 Documentation of the pattern mining process

Since patterns are generalizations of empirical cases they face the problem of induction. However, if we treat identified patterns as hypothetical design rules, we can test them in new projects. Therefore, it is suggested to follow an abductive reasoning in the pattern mining process: the pattern is induced in a process of critical thinking and tested deductively in new design cases. To understand the level of confidence and corroboration of a pattern, chapter 3.4.5 provided a justification for the induction of patterns, discussed the empirical and logical content of patterns, and proposed the empirical testing of identified patterns which are treated as hypothetical assumptions about morphological laws. The practical implication of this discussion has been that pattern descriptions should document the mining ground (variation of cases), the mining methods (validity of cases, confidence, objectivity), the known uses that induced a pattern (sample size and variation of cases), and the degree of corroboration (successful application of patterns to similar cases as evidence).

5.6.3 Future work

Which of the mining methods works best depends on the situation or context. This calls for describing the mining methods as patterns themselves. Hanmer (2012) has started writing patterns for pattern mining. These patterns are focussed on finding and documenting patterns. All of the methods described in chapter 3.4 could be described as patterns as well. This would help to understand their applicability, benefits and limits. Moreover, all methods of scientific inquiry could probably be described in the pattern format. Very often researchers use the inappropriate methods for their research questions. Patterns could help to clarify the contexts and conditions for the use of a method. Hoda, Noble & Marshall (2011) have collected some patterns to understand grounded theory in the context of computer science. In personal communication with Peter Baumgartner during an educational patterns workshop we agreed that it would be beneficial to have a pattern language for scientific research and writing.

5.7 Old wine in new bottles

Research Question RQ-7: What is novel and original in the pattern approach?

5.7.1 Original extraction of universal forms

We have argued that patterns are theories – ways of looking at things or getting insights to objects. Our discussion about partitioning structures has shown that there are always alternative ways to look at things. That is why we can say that patterns do provide new insights if they are an *original* way of looking at things. The implicate order of the world allows many different explications. A meaningful explication can provide novel understandings of proven designs.

5.7.2 The discovery of new regularities

There are two other reasons to see a pattern itself as original when it is described for the first time. First, we can actually discover new regularities or relations that have not been noticed before. For example, to find out that a good practice is not only performed in development team A but also in B is informative. It strengthens the confidence into the structure.

Moreover, by considering the same recurrent form we could discover new findings. For example, the recommended durations for an online training are rules of thumbs. But we could also run a study that quizzes participants about the durations they prefer. Such a study would provide original findings about the form that are novel. As we have seen, the term in-formation suggests that in the very structure of one form (or pattern) there is more “in” than we superficially see. By making observations of online trainings we find out which durations work – we are informed about the appropriate ranges of times. This information is an implicit property of the form of online trainings and has always been there – but we have to ask the right questions to obtain this information.

5.7.3 Research about known patterns

About any given pattern we can get more knowledge, more insights and information by further research. The evolution of patterns is due to novelty of insights – learning something about the pattern that was previously unknown. We have seen examples in chapter 4.4: The pattern content has changed because new technologies merge (multi-touch systems) or more experience is taken into account (the suggested implementation for PowerPoint cannot be recommended any more). Patterns can also be described with different levels of details and providing different amount of information about the pattern. Two pattern descriptions p1 and p2 may pull out different amounts of information. There can be many reasons for this differentiation. One is that pattern author team A is more informed than pattern author team B – for example, author team A may have more experience; thus it knows more about the properties of the pattern than author team B. Another reason for different amounts of information provided in a pattern description is that audiences are equipped with different expectations and degrees of prior knowledge. Later versions of the patterns of interactive graphics (see chapter 4.2.1) contained less technical implementation details because we identified end-users as the main target group to read the patterns. The reflection about the evolution of the pattern format (see chapter 4.4.2.1) is an example about experimenting and improving the pattern descriptions based on the feedback from Writers' Workshops and the shepherding process.

5.7.4 In search of an invariant

Very often similar practices have emerged independently. For example when we discovered that the same interaction types were used in different multimedia titles or the same activities were performed in online trainings, we found evidence that there is a kind of "necessity". The solution form was not in use accidently but for a reason. The given situation puts some demands to the designers. There may have been alternative solutions but given the fact that in one culture we share knowledge about similar tools, practices and problems, it is likely that two or more persons come up with the same solutions independently. The reason is that new solutions are always derived from existing knowledge. The "Eureka!" moment is a new understanding of a problem. However, this new understanding is based on connecting prior knowledge. That is why many discoveries and inventions have been made at the same time but "independently". Patterns are an attempt to capture those solutions that have been found multiple times. Harmonizing the multiple instances into one coherent view is an original and creative act.

5.7.5 Novelty of the pattern approach

The other kind of novelty that has been questioned is about the pattern approach in general. Indeed, many of the ideas of the pattern approach can be found elsewhere. In our discussion of the structure of patterns we have seen the importance of the different views that are taken by the pattern format. There are descriptions of methods, models, etc. that satisfy the requirements of a pattern description without being called patterns. But they are patterns nevertheless. We could find millions of patterns that are not appreciated as "design" patterns.

From that point of view the pattern approach is only a summary of what matters for good designs. It requires us to consider solutions as wholes and it offers a framework to reason about the appropriate level of abstraction and granularity, to always investigate the contexts, to discuss the interplay of forces instead of manipulating variables ceteris paribus, to consider the combination of patterns into larger wholes with emerging new properties, appreciate the complexity of solutions and handle it by meaningful decomposition, discuss problems and solutions on multiple levels, and acknowledge the consequences and side effects of an intervention. If none of these aspects is new, then the pattern approach is not new. The only difference might be that the pattern community integrates these aspects into one methodology.

5.7.6 Future work

This work has discussed when and to which extent patterns present new findings. However, we did not study how many of the existing pattern descriptions explicate this data. Future research could investigate pattern descriptions and compare the presented knowledge to other resources. This would show which information is new in the pattern documents. It would also reflect which information is additionally revealed due to the pattern format. By asking the right questions the pattern format might pull out more tacit knowledge from experts. For example, a teaching method described as a pattern might not be new but by explicating the context we get new information. To find such cases could be a new research project and provide evidence for the benefits of the pattern approach.

5.8 Misapplication of patterns

Research Question RQ-8: How to avoid misapplications of patterns?

5.8.1 Violation of pattern ethics

In section 2.4 we have highlighted the quality standards for patterns. Pattern ethics require authors to provide known uses and examples as evidence for the solutions. Patterns that are not whole are often refined or discarded in the process of shepherding and after Writers' Workshops. Pattern descriptions that are not discussed in this way often lack maturity. To use the pattern format to package untested ideas or get another publication is a misapplication of patterns at the author's side.

5.8.2 Patterns need to be implemented in the right way in the right context

There are at least two common sources of misapplication by users that make patterns fail: not understanding the context (the constraints and applicability) and to miss the difference between the pattern and its instantiation – to take the map for the territory. To implement a pattern – to follow a path on a map – is different to the theoretical structure. Usually to follow a path one requires further skills, e.g. good sense of navigation or climbing. Likewise, to follow a pattern in software design, programming and analytical skills are needed. To use a pedagogical method requires social skills to adapt it to the actual learning situation.

5.8.3 The success of e-learning tools and patterns depends on the context

Many of the potentials of e-learning and digital media are not used because the tools are not well known, used in the wrong contexts or in the wrong ways. A hammer is only a good tool if it is used for the right task and in the right way. Using a hammer in the wrong way could actually be very harmful. The same is true for digital media and e-learning tools: if we use a tool for the wrong situation the learning outcome might be worse. Yet it is not the tool to be blamed for its wrong application.

It is also wrong to make judgments about the value of a form without considering the context. This is major error in current empirical research where interventions are considered in isolation. An intervention can be highly beneficial in situation A, neutral in situation B, or worse in situation C. A general statement such as “intervention X has always positive or negative effects” is not justified unless it has been tested in all possible contexts.

5.8.4 Patterns can put demands to the context

In our discussion of pattern languages we have seen how patterns interplay to generate beneficial effects. In this interplay one pattern is the context of the other and vice versa. This implies that very often the context has to change as well in order to make a new tool, method or media type a success because they unfold their positive potential only in the right context. Once man invented the hammer he changed the way of building. E-Learning and social media tools change the way we learn – in other words an e-learning tool often requires a change of learning methods. Using old methods with new tools could be harmful. But if we use a new tool and apply new methods appropriately the outcome should be better.

As a result, the context description of a pattern should not only tell us when the pattern can be used but what is required to make the pattern a success. These requirements are sometimes found in the liabilities or counter-indication sections. The requirements could also be imperative in telling us what we have to do before we can use the pattern. The wide use of digital media can only be a success in a context where teachers are trained in new methods, the service infrastructure for hard- and software is working, and the systems are robust.

5.8.5 Future work

Chapter 3.4 has ended with the conclusion that papers about patterns should include information about the mining ground, the mining methods, the known uses that induced a pattern and the degree of corroboration (successful application of patterns after it has been written). An interesting research question is to investigate how many of the published pattern papers include this information and whether this correlates with the quality of the patterns. Future research also needs to continuously update this information for the patterns presented in this thesis.

The organization boards of the pattern conferences also discuss how to improve the quality of pattern submissions. At EuroPLoP there is now an additional quality gate for papers to be selected and published in the ACM online proceedings (based on recommendations from workshop leads and a third round of reviews by members of the program committee). This should encourage authors to improve their papers taking the feedback from the Writers’ Workshops into account.

5.9 Outlook

The previous section presented emerging research questions based on the conclusions and findings of this thesis. Following the research framework from Holz et al. (2006) we have asked “Have we achieved our goal?” and are now able to succeed with the question “What do we want to achieve next?” Besides the derived research questions the author considers three pattern mining projects for future work.

The short-term project is to mine patterns for creative thinking (Kohls, 2012b). Creative thinking and the development of new ideas or solutions is complimentary to the knowledge captured in patterns. There are many descriptions of creativity methods and tools available. What motivates the description as patterns is the generalization of similar methods, the reasoning for the actual form in terms of forces, and the contextualization and connection of the methods or tools. Very often a specific method implies other methods to follow up or it can be combined with other methods. A pattern language captures such relations

The mid-term project is to describe more patterns for blended learning. These activities will be coordinated with research fellows who collaborate via educationalpatterns.org. The findings of this thesis motivate to create a network of many smaller languages rather than isolated patterns or one large pattern language. e-teaching.org has started to write some of its articles in the pattern format and the author plans to contribute articles in the future.

The long-term project is to create a pattern language for pattern mining. This language will represent many of the findings of this thesis but capture them in the pattern format itself. For example, all of the methods for pattern mining could be re-written as patterns.

References

Alexander, C. (1964). *Notes on the synthesis of form.* Cambridge: Harvard University Press.

Alexander, C. (1975). *The Oregon experiment*. New York: Oxford University Press.

Alexander, C. (1979). *The Timeless Way of Building*. New York: Oxford University Press.

Alexander, C. (2002a). *The Nature of Order, Book 1. The phenomenon of life.* Berkeley, Calif: Center for Environmental Structure.

Alexander, C. (2002b). *The Nature of Order, Book 2. The phenomenon of life.* Berkeley, Calif: Center for Environmental Structure.

Alexander, C., Ishikawa, S., & Silverstein, M. (1968). *A pattern language which generates multi-service centers.* University of California, Berkely: Center for environmental structure.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A pattern language*. New York, USA: Oxford University Press.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge: Harvard University Press.

Anthony, D.L. (1996). Patterns for Classroom Education. In J. Vlissides, C.O. Coplien, & N.L. Kerth, (Eds.), *Pattern Languages of Program Design 2* (pp. 391-406). Reading, Mass: Addison-Wesley.

Appelton, B. (2000). *Patterns and Software: Essential Concepts and Terminology.*
http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html. (1.7.2009)

Aristotle(1957). In W. Jaeger (Ed), *Greek text: Aristotelis Metaphysica.* Oxford Classical Texts. Oxford: Oxford University Press.

Ash, M. G. (1995). *Gestalt psychology in German culture, 1890-1967: Holism and the quest for objectivity.* Cambridge studies in the history of psychology. Cambridge: Cambridge University Press.

Ashby, W. R. (1956). *An Introduction to Cybernetics.* London: Chapman & Hall.

Avgeriou, P., Retalis, S., & Papasalouros (2004). Patterns for Designing Learning Management Systems. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp.115-132). Konstanz: Universitätsverlag Konstanz.

Baggetun, R., Rusman, E., & Poggi, C. (2004). Design patterns for collaborative learning: From practice to theory and back. In *Proceedings of International Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 2493-2498). Lugano, Switzerland.

Ball, P. (2009). *Shapes - Nature's patterns: A tapestry in three parts.* Oxford: Oxford University Press.

Bartlett, F. C. (1932). *Remembering: An experimental and social study.* Cambridge: Cambridge University Press.

Von Baeyer, H. C. (2004). *Information: The new language of science.* London: Phoenix.

Barr, B., Biddle, R., Noble, J. (2004). Interface Ontology: Creating a Physical World for Computer Interfaces. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp.1-18). Konstanz: Universitätsverlag Konstanz.

Bauer, R., & Baumgartner, P. (2011). A First Glimpse at the Whole: Christopher Alexander's Fifteen Fundamental Properties of Living Centers and Their Implication for Education. In C.Kohls, & J. Wedekind, (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions* (pp. 272-283). Hershey: Information Science Pub.

Bauer, R., & Reinmann, G. (2010). *Förderung wissenschaftlicher Schreibkompetenz durch Writers´ Workshops (Forschungsnotiz Nr. 4).* München: Universität der Bundeswehr München, Lehren und Lernen mit Medien. URL: http://lernen-unibw.de/sites/default/files/Forschungsnotiz_2010_04.pdf (1.8.2011).

Baumgartner, P. (2006). Unterrichtsmethoden als Handlungsmuster - Vorarbeiten zu einer didaktischen Taxonomie für E-Learning. In M. Mühlhäuser, G. Rößling, & R. Steinmetz (Eds.), *DeLFI 2006, 4. e-Learning Fachtagung Informatik* (pp. 51-62). Darmstadt: Gesellschaft für Informatik e.V.

Baumgartner, P. (2011). *Taxonomie von Unterrichtsmethoden: Ein Plädoyer für didaktische Vielfalt.* Münster: Waxmann.

Beck, K., Coplien, J. O., Crocker, R., Dominick, L., Meszaros, G., Paulisch, F., & Vlissides, J. (1998). Industrial Experience with Design Patterns. In L. Rising (Ed.), *The Pattern Handbook* (pp.203-228).Cambridge: Cambridge University Press.

Beck, K., & Cunningham, W. (1987). Using Pattern Languages for Object-Oriented Programs. *Technical Report CR-87-43*, Tektronix, Inc. OOPSLA'87 workshop on Specification and Design for Object-Oriented Programming.

Beck, K, Nilsson, N., & Marinescu, F. (2008). *Kent Beck on Implementation Patterns.* Interview with Kent Beck. http://www.infoq.com/interviews/beck-implementation-patterns (1.12.2011)

Bergin, B. (2001). Fourteen Pedagogical Patterns. In M. Devos, & A. Rüping (Eds.), *EuroPLoP 2000. Proceedings of the 4th European Conference on Pattern Languages of Programs* (pp.1-40). Konstanz: Universitätsverlag Konstanz.

Bergin, B. (2004). Two Pedagogical Patterns for Course Design. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (p.133-138). Konstanz: Universitätsverlag Konstanz.

Berkun., S. (2007). *The myths of innovation.* Sebastopol, CA: O'Reilly.

Bienhaus, D. (2004). Some Pedagogical Patterns from a System-Centred Approach. In D.Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp. 139-164). Konstanz: Universitätsverlag Konstanz.

Bienhaus, D. (2009). Some Patterns for Innovation. In A. Kelly, & M. Weiss (Eds.). *EuroPLoP 2009. Proceedings of the 14th European Conference on Pattern Languages of Programs.*

Bjørner, D. (2006*). Software Engineering 1 - Abstraction and Modelling.* Berlin, Heidelberg: Springer.

Bobik, J., & Sayre, K. M. (1963). Pattern Recognition Mechanisms and St. Thomas' Theory of Abstraction. In *Revue Philosophique de Louvain.* Troisième série, Tome 61, 69. 24-43.

Bohm, D. (1981). *Wholeness and the implicate order*. London: Routledge & Kegan Paul.

Bohm, D., & Factor, D. (1985). *Unfolding meaning: A weekend of dialogue with David Bohm.* London: Routledge.

Booch, G. (1998). Patterns. In L. Rising (Ed.) *The Pattern Handbook.* Cambridge: Cambridge University Press.

Borchers, J. (2001). *A pattern approach to interaction design*. Chichester: Wiley.

Bortoft, H. (1996). *The wholeness of nature: Goethe's way toward a science of conscious participation in nature. Renewal in science.* Hudson, N.Y: Lindisfarne Press.

Bortz, J., & Döring, N. (2002). *Forschungsmethoden und Evaluation: Für Human- und Sozialwissenschaftler*. Berlin: Springer.

Bourdieu, P. (1994). Sozialer Raum, symbolischer Raum. Reprinted in J. Dünne, & S. Günzel (2006), *Raumtheorie. Grundlagentexte aus Philosophie und Kulturwissenschaften*. Frankfurt a.M.: Suhrkamp.

Bricout, V., Heliot, D., Cretoiu, A., Yang, Y., Simien, T., & Hvatum, L. (2005). Patterns for Managing Distributed Product Development Teams. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (p. 109-122). Konstanz: Universitätsverlag Konstanz.

Brown, V. A. (2008). A pattern for collective social learning. In T. Schümmer, & A. Kelly (Eds.*), EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs.*

Buchholz, K. (2006). *Ludwig Wittgenstein*. Campus Einführungen. Frankfurt: Campus.

Buschmann, F., Henney, K., & Schmidt, D.C. (2007). *Pattern-oriented software architecture. Volume 5: On patterns and Pattern Languages.* West Sussex: John Wiley & Sons.

Buschmann, F.; Meunier, R; Rohnert, H., Sommerlad, P. & Stal, M. (1996*). Pattern-*
Oriented Software Architecture – A System of Patterns. West Sussex, England: Wiley and Sons.

Cassirer, E. (1931). Mythischer, ästhetischer und theoretischer Raum. Reprinted in J. Dünne, & S. Günzel (2006*), Raumtheorie. Grundlagentexte aus Philosophie und Kulturwissenschaften*. Frankfurt a.M.: Suhrkamp.

Chaitin, G.J. (1999): *The Unknowable*. Berlin: Springer.

Chang, D., Dooley, L., & Touvinen, J. E. (2002): Gestalt Theory in Visual Screen Design: A New look at an Old Subject. *ACM International Conference Proceeding Series.* 26.

Chen, N., & Rabb, M. (2009). A Pattern Language for Screencasting. In *Proceedings of the 16th Conference on Pattern Languages of Programs*. Chicago, IL: ACM.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*. 5. 121-152.

Clark, R. C., & Mayer, R. E. (2003). *E-Learning and the science of instruction: proven guidelines for consumers and designers of multimedia learning.* San Francisco, CA: Jossey-Bass/Pfeiffer.

Coldewey, J. (1998). User Interface Software. In *Proceedings of the 5th Conference on Pattern Languages of Programs.* http://hillside.net/plop/plop98/final_submissions/P13.pdf (14.7.2011).

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology.* 79. 4. 347-362.

Cooper, J. W. (2000). *Java design patterns: a tutorial*. Reading, Mass: Addison-Wesley.

Coplien, J. O. (1992). *Advanced C++ programming styles and idioms*. Reading, Mass: Addison-Wesley Pub.

Coplien, J. O. (1996). *Software Patterns*. New York: SIGS Books & Multimedia

Coplien, J.O. (1998a). Space: The Final Frontier. *C++ Report*. 10(3). 11-17.

Coplien, J.O. (1998b). Setting the Stage. In L. Rising (Ed.). *The Pattern Handbook* (pp.87-96), Cambridge: Cambridge University Press.

Coplien, J.O. (1998c). A Generative Development – Process Pattern Language. In L. Rising (Ed.), *The Pattern Handbook* (pp.243-300). Cambridge: Cambridge University Press.

Coplien, J.O. (1998d). Software Design Patterns: Common Questions and Answers. In L. Rising (Ed.), *The Pattern Handbook* (pp. 311-320). Cambridge: Cambridge University Press.

Coplien, J.O. (1998e). Software Development as Science, Art, Engineering. In L. Rising (Ed.), *The Pattern Handbook* (pp.321-332). Cambridge: Cambridge University Press.

Coplien, J.O. (2001). The future of language: Symmetry or broken symmetry? *In Proceedings of VS Live 2001*. San Francisco, California.

Coplien, J. O. (2004). The Culture of Patterns. *Computer Science and Information Systems.* 1(2).

Coplien, J. O., & Appleton, B. (1997). On the Nature of Order. Notes by Brad Appelton on a Presentation given by James O. Coplien to the Chicago Patterns Group on The Nature of Order.

Coplien, J. O., & Woolf, B. (2000). A Pattern Language for Writers' Workshops. In B. Foote, N. Harrison, & H. Rohnert (Eds.), *Pattern Languages of Program Design 4* (pp. 557-584). Addison Wesley, Reading, MA, 2000.

Coplien, J.O., & Zhao, L. (2001). *Symmetry Breaking in Software Patterns.* Springer Lecture Notes in Computer Science. LNCS 2177. 37-54. Berlin / Heidelberg: Springer.

Corfman, R. (1998). An Overview of Patterns. In L. Rising (Ed.). *The Pattern Handbook* (pp.87-96), Cambridge: Cambridge University Press.

Cress, U., & Kimmerle, J. (2008). A systemic and cognitive view on collaborative knowledge building with wikis. *International Journal of Computer-Supported Collaborative Learning.* 3(2). 105-122.

Crossman, E. R. F. W. (1959). A Theory of the acquisition of speed-skill. *Ergonomics.* 2. 153-166

Csikszentmihalyi, M. (1996). *Creativity: Flow and the psychology of discovery and invention.* New York: HarperCollins Publishers.

Cummins, D. D. (1992). Role of analogical reasoning in the induction of problem categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition.* 18. 1103-1124

Cybulski, J.L., & Linden, T. (1998). Composing Multimedia Artifacts for Reuse. In *Proceedings of the 5th Conference on Pattern Languages of Programs.* http://hillside.net/plop/plop98/final_submissions/P38.pdf (14.7. 2011).

Dawkins, R. (1989). *The selfish gene.* Oxford: Oxford University Press.

DeLano, D. E. (1998). Patterns Mining. In L. Rising (Ed.), *The Pattern Handbook* (pp.87-96). Cambridge: Cambridge University Press.

Derntl, M. (2006). *Patterns for person centered e-learning.* Berlin: AKA.

Descartes, R. (1644). Über die Pinzipiel der materiellen Eigenschaften. Reprinted in J. Dünne, & S. Günzel (2006), *Raumtheorie. Grundlagentexte aus Philosophie und Kulturwissenschaften.* Frankfurt a.M.: Suhrkamp.

Detel, W. (2005). *Aristoteles.* Grundwissen Philosophie. Leipzig: Reclam.

Dillenbourg, P. (2002) Over-Scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed.), *Three worlds of CSCL. Can we support CSCL.* 61-91. Heerlen: Open Universiteit Nederland.

Dillenbourg, P., & Jermann, P. (2007). Designing integrative scripts. In F. Fischer, I. Kollar, J. Mandl, & J.M. Haake (Eds.), *Scripting computer-supported collaborative learning: Cognitive, computational, and educational perspectives (*pp. 278-301). New York: Springer.

Dimitriadis, Y, Goodyear, P., & Retalis, S. (Eds.) (2009). Design Patterns for Augmenting E-Learning Experiences. *Computers in Human Behavior.* Special Issue on Design Patterns for Augmenting E-Learning Experiences. 25 (5).

Döbeli-Honegger, B., & Notari, M. (2011). Visualizing learning processes using Didactic Process Maps. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions.* Hershey: Information Science Pub.

Dong, J., Zhao, Y., & Peng, T. (2009). A Review of Design Pattern Mining Techniques. In *International Journal of Software Engineering and Knowledge Engineering.* 19 (6). 823-855.

Duncker, K. (1945). On problem-solving. *Psychological Monographs.* 58 (270). 1-113.

Dünne, J., & Günzel, S. (2006). *Raumtheorie. Grundlagentexte aus Philosophie und Kulturwissenschaften.* Frankfurt a.M.: Suhrkamp.

Eckblad, G. (1981). *Scheme theory: a conceptual framework for cognitive-motivational processes.* London: Academic Press.

Eckstein, J. (1999). Incremental Role Play. In J. Coldewey, & P. Dyson (Eds.), *EuroPLoP 1998. Proceedings of the 3rd European Conference on Pattern Languages of Programs* (pp.53-58). Konstanz: Universitätsverlag Konstanz.

Eckstein, J. (2001). Learning to Teach and Learning to Learn - Pedagogical and Social Issues in Education. In M. Devos, & A. Rüping (Eds.), *EuroPLoP 2000. Proceedings of the 4th European Conference on Pattern Languages of Programs* (pp.75-86). Konstanz: Universitätsverlag Konstanz.

Eckstein, J. (2003). Patterns for Active Learning. In A. O'Callaghan, J. Eckstein, & C. Schwanninger (Eds.), *EuroPLoP 2002. Proceedings of the 4th European Conference on Pattern Languages of Programs.* Konstanz: Universitätsverlag Konstanz.

Eckstein, J., Manns, M.L., Sharp, H., & Sipos, M. (2004). Teaching from Different Perspectives. In D. Schütz, & K. Marquardt (Eds.). *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp. 165-182). Konstanz: Universitätsverlag Konstanz.

Eckstein, J., Marquardt, K., Manns, M., and Wallingford, E. (2001). Patterns for experiential learning. In Rüping, A., Eckstein, J., and Schwanninger, C. Proceedings of the 6th European Conference on Pattern Languages of Programs Konstanz: UVK, Universitäts-Verlag Konstanz.

Edelman, G. M. (2006). *Second nature: brain science and human knowledge.* New Haven: Yale University Press.

Ehrenfels (1890). Über Gestaltqualitäten. *Vierteljahrsschrift für wissenschaftliche Philosophie*. 14. 249-292.

Erdmann, B. (1892). *Logische Elementarlehre.* Halle A. S.: Verlag Max Niemeyer.

Erickson, T. (2000). Lingua francas for design: Sacred places and pattern languages. In *Designing Interactive Systems 2000*. 357–368. Brooklyn, NY: ACM Press.

Feyerabend, P., & Terpstra, B. (1999). *Conquest of abundance: A tale of abstraction versus the richness of being.* Chicago: University of Chicago Press.

Feynman, R. (1974): Cargo Cult Science. *Engineering and Science.* 37:7. 10–13.

Fine, K. (2002). *The limits of abstraction.* Oxford: Oxford University Press.

Finlay, J., Gray, J., Falcone, I., Hensman, J., Mor, Y., & Warburton, S. (2009) *Planet: Pattern Language Network for Web 2.0 in Learning.*

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Ed.). *Categories of human learning*. New York: Academic Press.

Flechsig, K.-H. (1996). *Kleines Handbuch didaktischer Modelle*. Eichenzell: Neuland, Verl. für Lebendiges Lernen.

Flick, U. (1998). *An introduction to qualitative research*. London: Sage.

Floridi, L. (2010). *Information: A very short introduction*. Oxford: Oxford University Press.

Focault, M. (1967). Von anderen Räumen. Reprinted in J. Dünne, & S. Günzel (2006), *Raumtheorie. Grundlagentexte aus Philosophie und Kulturwissenschaften*. Frankfurt a.M.: Suhrkamp.

Foote, B. (1997) In R.C. Martin, D. Riehle, & F. Buschmann (Eds.), *Pattern Languages of Program Design 3*. Reading: Addisson Wesley.

Foote, B., & Yoder, J. (1997). Big Ball of Mud. *Fourth Conference on Patterns Languages of Programs (PLoP '97/EuroPLoP '97).* Monticello, Illinois, September 1997

Forschungswerkstatt (2009). *Forschungswerkstatt Didaktische Entwurfsmuster.* Hosted by P. Baumgartner http://www.peter.baumgartner.name/Members/baumgartner/news/forschungswerkstatt-didaktische-entwurfsmuster. Commenty protocol by C. Kohls. Vienna: 2009.

Freeman, E., Freeman, E., Sierra, K., & Bates, B. (2004). *Head First design patterns*. Sebastopol, CA: O'Reilly.

Gabriel, R. P. (1996). *Patterns of software: Tales from the software community*. New York: Oxford University Press.

Gabriel, R. P. (1998). The Failure of Pattern Languages. In L. Rising, (Ed.). *The Pattern Handbook* (pp.333-344). Cambridge: Cambridge University Press.

Gabriel, R. P. (2008a). *Writers' Workshops As Scientific Methodology*. http://dreamsongs.com/Essays.html (01.08.2008)

Gabriel, R. P. (2008b). Designed as designer. *SIGPLAN Not*. 43, 10 (Oct. 2008), 617-632. DOI= http://doi.acm.org/10.1145/1449955.1449813.

Gabriel, R. P. (2002). *Writers workshops and the work of making things: Patterns, poetry*. Boston, Mass.: Addison-Wesley.

Gabriel, R.P., & Goldman, R. (1999). In *Proceedings of the 6th Conference on Pattern Languages of Programs*. http://hillside.net/plop/plop99/proceedings/gabriel/JiniCommunityPL_excerpt.pdf

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: Addison-Wesley.

Garrido, A., Rossi, G., & Schwabe, D. (1997). Pattern Systems for Hypermedia. *Proceedings of the 4th Conference on Pattern Languages of Programs.* http://hillside.net/plop/plop97/Proceedings/garrido.pdf (1.8.2011)

Gentner, D., & Stevens A.L. (1983). *Mental models.* Hillsdale, NJ: Lawrence Erlbaum Associates

Georgiadi, S., Retalis, S., & Georgiakakis, P. (2008). Design patterns for the announcements mechanism about courses and events in a Learning Management System. In T. Schümmer, & A. Kelly (Eds.). *EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs.*

Ghyka, M. C. (1977). *The geometry of art and life*. New York: Dover Publications.

Gick, M.L., and Holyoak, K.J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*. 15. 1-38.

Ginnis, P. (2002). *The teacher's toolkit: Raise classroom achievement with strategies for every learner*. Bancyfelin, Carmarthen, Wales: Crown House Pub.

Gleick, J. (2011). *The information: A history, a theory, a flood.* New York: Pantheon Books.

Goldstein, E. Bruce. (2009). *Sensation and Perception*. Wadsworth Pub.

Goodman, N. (1976). *Language of art: An approach to a theory of symbols*. Indianapolis, Ind: Hackett Publishing Co.

Goodyear, P. (2005). Educational design and networked learning: Patterns, pattern languages and design practice. *Australasian Journal of Educational Technology*. 21(1). 82-101.

Goodyear, P., de Laat, M., & Lally, V. (2006). Using pattern languages to mediate theory-praxis conversations in design for networked learning. *ALT-J, Research In Learning Technologies*. 14. 211-223. Association for Learning Technologies.

Gordon, I. E. (1989). *Theories of visual perception*. Chichester: Wiley.

Grenander, U. (1996). *Elements of pattern theory*. Johns Hopkins studies in the mathematical sciences. Baltimore: Johns Hopkins University Press

Haase, M. (2006) Patterns for Leading Effective and Efficient Meetings. In A. Longshaw, & U. Zdun (Eds.), *EuroPLoP 2005. Proceedings of the 15th European Conference on Pattern Languages of Programs* (pp.751-759). Konstanz: Universitätsverlag Konstanz.

Hadjisimou, M., Tzanavari, A. (2007). Migrating to e-Learning in Secondary Education. In U. Zdun, & L. Hvatum (Eds.), *EuroPLoP 2006. Proceedings of the 11th European Conference on Pattern Languages of Programs* (pp.669-674). Konstanz: Universitätsverlag Konstanz.

Hall, A.D., & Fagen, R.E. (1956). *Definition of Systems*. General Systems Yearbook. 1. 18-28.

Hanmer, R. (2007). *Patterns for fault tolerant software*. Chichester, England: John Wiley

Hanmer, R. (2012). Pattern Mining Patterns. *19th Pattern Languages of Programs conference.* Writers' Workshop version. Tuscon, Arizona.

Hanson, N.R. (1958). *Patterns of Discovery*. Cambridge.

Harrer, Andreas (2002). Nutzung von Software-Mustern zur Entwicklung von intelligenten Lehr-/Lernsystemen, In *LLA 02 -- Tagungsband der GI-Workshop-Woche "`Lehren - Lernen - Adaptivität"'*. Hannover, 2002.

Harrer, A., & Martens, A. (2005). Ansatz zur Definition einer Mustersprache für Lehr-/Lernsysteme. In: *Proceedings der Tagung: DeLFI, 3. Deutsche e-Learning Fachtagung der Gesellschaft für Informatik*, Rostock, Germany, 13.-18. September 2005, p. 177-188

Harrison, N.B. (1998). Potential Pattern Pitfalls, or How to Jump on the Patterns Bandwagon Without the Wheels Coming Off. In L. Rising (Ed,). *The Pattern Handbook* (pp.87-96). Cambridge: Cambridge University Press.

Harrison, N.B. (2006a). The Language of Shepherding. In D. Manolescu, M. Völter, & J.
Noble (Eds.). *Pattern Languages of Program Design 5*. Boston: Addison-Wesley.

Harrison, N.B. (2006b). Advanced Pattern Writing. Patterns for Experienced Pattern Authors. In D. Manolescu, M. Völter, & J. Noble (Eds.), *Pattern Languages of Program Design 5*. Boston: Addison-Wesley.

Harrison, N.B. & Coplien, J. O. (2002). Pattern Sequences. In A. Rüping, J. Eckstein, & C. Schwanninger (Eds.), *Proceedings of the 6th European Conference on Pattern Languages of Programs* (pp. 549-550). Konstanz: Universitätsverlag Konstanz.

Harrison, N.B., & Coplien, J. O. (2005). *Organizational patterns of agile software development*. Upper Saddle River, NJ: Pearson Prentice Hall.

Hesse, F. W. (1991). *Analoges Problemlösen: eine Analyse kognitiver Prozesse beim analogen Problemlösen*. Fortschritte der psychologischen Forschung, 8. Weinheim: Psychologie Verlags Union.

Hofstadter, D. R. (1980). *Gödel, Escher, Bach: An eternal golden braid.* New York: Vintage Books.

Hola, R., Noble, J., & Marshall, S. (2011). Grounded Theory for Geeks. *18th Pattern Languages of Programs conference*. Writers' Workshop version. Portland, Oregon.

Holland, J. H. (2000). *Emergence: From chaos to order*. Oxford: Oxford University Press.

Holz, H.J., Applin, A., Haberman, B., Joyce, D., Purchase, H., & Reed, C. (2006). Research methods in computing: what are they, and how should we teach them? *SIGCSE Bull*. 38, 4. 96-114.

Holzner, S. (2006) *Design Patterns For Dummies*. Hoboken: Wiley Publishing.

Homsky, O. (2004). More Patterns for Group Leadership. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs*. (pp. 183-198). Konstanz: Universitätsverlag Konstanz.

Homsky, O. (2005). Group Leadership at Events. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs*. (pp. 215-224). Konstanz: Universitätsverlag Konstanz.

Homsky, O., & Raveh, A. (2006). Patterns for Human Interaction in Software Maintainability. In A. Longshaw, & U. Zdun (Eds.), *EuroPLoP 2005. Proceedings of the 15th European Conference on Pattern Languages of Programs* (pp. 531-538). Konstanz: Universitätsverlag Konstanz.

Hughes, J., Rodden, T., Rouncefield, M.and Viller, S. (2000). Patterns of home life: Informing design for domestic environments. *Personal Technologies Special Issue on Domestic Personal Computing*, 4:25–38.

Hume, D., & Buckle, S. (2007). *An enquiry concerning human understanding and other writings*. Cambridge texts in the history of philosophy. Cambridge: Cambridge University Press.

Iacob, C. (2011). A Design Pattern Mining Method for Interaction Design. *EICS11*.

Iba, T. (2012). A Pattern Language for Designing Workshop to Introduce a Pattern Language. *18th European Pattern Languages of Programs conference*. Writers' Workshop version. Irsee, Bavaria.

Iba, T., & Isaku, T (2012). Holistic Pattern-Mining Patterns. *19th Pattern Languages of Programs conference.* Writers' Workshop version. Tuscon, Arizona.

Ingstrup, M. (2004). Interaction Widget. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp. 199-206). Konstanz: Universitätsverlag Konstanz.

Johnson, R. (1998a). How Patterns Work in Teams. In L. Rising (Ed.), *The Pattern Handbook* (pp.87-96).Cambridge: Cambridge University Press.

Johnson, R. (1998b). Patterns and Frameworks. In L. Rising (Ed.), *The Pattern Handbook* (pp.375-382).Cambridge: Cambridge University Press.

Johnson, S. (2002). *Emergence: The connected lives of ants, brains, cities, and software*. New York: Touchstone.

Kaiser, S., Kohls, C., & Windbrake, T. (2002). Javanti - Using Tcl to design interactive eLearning materials. *9th Annual Tcl/Tk Conference*. Vancouver, Canada.

Kant, I., & Erdmann, B. (1880). *Immanuel Kant's Kritik der Urtheilskraft*. Leipzig: Voss.

Kaschek, R. (2004). A little theory of abstraction. In B. Rumpe, & W. Hesse (eds.) *Modellierung 2004: Proceedings zur Tagung*. GI Lecture Notes in Informatics, P-45.

Kavanagh, M. (2005). Towards a Pattern Language For Business Process Modelling. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp. 489-506). Konstanz: Universitätsverlag Konstanz.

Kelly, A. (2005). Business Strategy Design Patterns. The Porter Patterns. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs.* (pp. 1-26). Konstanz: Universitätsverlag Konstanz.

Kelly, A. (2006). A few more business patterns. In A. Longshaw, & U. Zdun (Eds.), *EuroPLoP 2005. Proceedings of the 15th European Conference on Pattern Languages of Programs* (pp. 261-278). Konstanz: Universitätsverlag Konstanz.

Kelly, A. (2007). More patterns for Technology Companies Product Development. In L. Hvatum, & T. Schümmer (Eds.), *EuroPLoP 2007. Proceedings of the 12th European Conference on Pattern Languages of Programs* (pp. 173-202). Konstanz: Universitätsverlag Konstanz.

Kelly, A. (2008). Business Patterns for Product Development. In T. Schümmer, & A. Kelly (Eds.), *EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs*. Irsee, Bavaria.

Kelly, G. M. (1982). Basic concepts of enriched category theory. *London Mathematical Society lecture note series*. 64. Cambridge: Cambridge University Press.

Kerth, N.L., Cunningham, W. (1997): Using Patterns to Improve Our Architectural Vision. *IEEE Software.* 14(1). 53-59.

King, D. B., & Wertheimer, M. (2005). *Max Wertheimer & Gestalt theory*. New Brunswick: Transaction Publishers.

Koenig, A. (1998). *Patterns and Antipatterns*. In L. Rising (Ed.). *The Pattern Handbook*. Cambridge: Cambridge University Press

Kogut, P. (1998). Design Reuse: Chemical vs. Software Engineering. In L. Rising (Ed.) *The Pattern Handbook.* Cambridge: Cambridge University Press.

Kohls, C. (2001*). Erstellung eines Computergestützten Tutoriums und Erarbeitung eines Einsatzkonzeptes für die Lehr- und Lernanwendung "jtap" unter didaktischen, wirtschaftlichen und technischen Gesichtspunkten*. Diploma Thesis.

Kohls, C. (2004). *Conception and Implementation of a Visual Language to create Interactive Presentations without Programming*. Master Thesis.

Kohls, C. (2009a). A Pattern Language for Online Trainings. In A. Kelly, & M. Weiss (Eds.), *EuroPLoP 2009. Proceedings of the 14th European Conference on Pattern Languages of Programs.* Irsee, Bavaria.

Kohls, C. (2009b). E-Learning Patterns - Nutzen und Hürden des Entwurfsmuster-Ansatzes. In N. Apostulopoulos, H. Hoffmann, V. Mansmann, & A. Schwill (Eds.), *E-Learning 2009. Lernen im digitalen Zeitalter* (pp. 61-72). Münster: Waxmann.

Kohls, C. (2010a). A pattern collection for interactive information graphics. In M. Weiss, & P. Avgeriou (Eds.), *EuroPLoP 2010. Proceedings of the 15th European Conference on Pattern Languages of Programs*.

Kohls, C. (2010b). The Structure of Patterns. *PLoP 2010 - 17th Pattern Languages of Programs conference*. Writers' Workshop version. Reno, Nevada.

Kohls, C. (2011a). The Structure of Patterns - Part II: Qualities. *PLoP 2011 - 18th Pattern Languages of Programs conference.* Writers' Workshop version. Portland, Oregon.

Kohls, C. (2011b). Patterns as an analysis framework to document and foster excellent e-learning designs. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions* (pp.19-40). Hershey: Information Science Pub.

Kohls (2012a). Erprobte Einsatzszenarien für Interaktive Whiteboards. In G.Csanyi, F. Reichl, A.Steiner (Eds.), *Digitale Medien. Werkzeuge für exzellente Forschung und Lehre* (pp. 187-197). Münster: Waxmann.

Kohls (2012b). Patterns for Creative Thinking. *19th Pattern Lnaguages of Programs conference*. Writers' Workshop version. Tuscon, Arizona.

Kohls, C. (2012c). Patterns and Paths - Introducing the path metaphor. *EuroPLoP 2012 - 18th European Pattern Languages of Programs conference.* Writers' Workshop version. Irsee, Bavaria.

Kohls, C., & Panke, S. (2009). Is that true? Thoughts on the epistemology of patterns. *Proceedings of the 16th Conference on Pattern Languages of Programs*. Chicago: ACM.

Kohls, C., & Scheiter, K. (2008). The relation between design patterns and schema theory. *Proceedings of the 2008 conference on Pattern Languages of Programs (PLoP).* Nashville, Tennessee: ACM.

Kohls, C., & Uttecht, J. G. (2009). Lessons learnt in mining and writing design patterns for educational interactive graphics. *Computers in Human Behavior*. 25 (5). 1040-1055.

Kohls, C., & Wedekind, J. (2008). Die Dokumentation erfolgreicher E-Learning-Lehr-/Lernarrangements mit didaktischen Patterns. In S. Zauchner, P. Baumgartner, E. Blaschitz, & A. Weissenbäck (Eds.), *Offener Bildungsraum Hochschule: Freiheiten und Notwendigkeiten*. pp. 217-227. Münster: Waxmann Verlag.

Kohls, C., & Wedekind, J. (2009). E-Learning Patterns Workshop. Tübingen, 04.-06.03.2009. http://www.iwm-kmrc.de/workshops/e-learning-patterns/ (01.05.2009)

Kohls, C., & Wedekind, J. (2011a). *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions*. Hershey: Information Science Pub.

Kohls, C., & Wedekind, J. (2011b). Perspectives on patterns. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions* (pp.2-18). Hershey: Information Science Pub.

Kohls, C., & Windbrake, T. (2003). Autorenwerkzeug Javanti. In *Proceedings of 1. Fachtagung e-Learning der Gesellschaft für Informatik (GI). DELFI.* München, 2003

Kohls, C., & Windbrake, T. (2006). Towards a Pattern Language for Interactive Information Graphics. *Proceedings of the 2006 conference on Pattern languages of programs*. Portland, Oregon: ACM.

Kohls, C., & Windbrake, T. (2007). Where to go and what to show: more patterns for a pattern language of interactive information graphics. *Proceedings of the 2007 conference on Pattern languages of programs.* Champaign-Urbana, IL: ACM.

Kohls, C., & Windbrake, T**.** (2008a). Moving objects – More patterns for a pattern language of interactive information graphics. In L. Hvatun, & T. Schümmer (Eds.), *Proceedings of the 12th European Conference on Pattern Languages of Programs, 2007: EuroPLoP '07* (pp. 321-343). Konstanz: Universitätsverlag Konstanz.

Kohls, C., & Windbrake, T. (2008b). Turning me on, turning me off (Writer's Workshop version). *13th European Conference on Pattern Languages of Programs.* 2008. http://hillside.net/europlop/europlop2008/submission/schedule.cgi.

Köhne, S. (2005). *Didaktischer Ansatz für das Blended Learning: Konzeption und Anwendung von Educational Patterns.* Dissertation, Universität Hohenheim.

Kolfschoten, G.L., & Santanen, E.L. (2007). Reconceptualizing Generate ThinkLets: the Role of the Modifier. *Hawaii International Conferenceon System Science*. Waikoloa: IEEE Computer Society Press.

Kollar, I., Fischer, F., & Hesse, F.W. (2006). Collaboration scripts – a conceptual analysis. *Educational Psychology Review*. 18(2). 75-86.

Köppe, C. (2011). A Pattern Language for Teaching Design Patterns. *PLoP 2011 - 18th Pattern Languages of Programs conference*. Writers' Workshop version. Portland, Oregon.

Köppe, C., Nijsten, M. (2012a). A Pattern Language for Teaching in a Foreign Language - Part 1. *18th European Pattern Languages of Programs conference*. Writers' Workshop version. Irsee, Bavaria.

Köppe, C., Nijsten, M. (2012b). A Pattern Language for Teaching in a Foreign Language - Part 2. *19th Pattern Languages of Programs conference*. Writers' Workshop version.Tuscon, Arizona.

Kortenkamp, U., & Blessing, A. M. (2011). VideoClipQuests as an E-Learning Pattern. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions*. Hershey: Information Science Pub.

Laurillard, D., & Ljubojevic, D. (2011). Evaluating Learning Design through the Formal Representation of Pedagogical Patterns. In C. Kohls & J., Wedekind (Eds). *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions.* Hershey: Information Science Pub.

Lea, D. (1994). Christopher Alexander: An Introduction for Object-Oriented Designers.
ACM Software Engineering Notes.

Leitner, H. (2007*). Mustertheorie: Einführung und Perspektiven auf den Spuren von Christopher Alexander*. Graz: Nausner & Nausner.

Li, M., & Vitanyi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications.* 2nd edition. New York: Springer.

Lidwell, W., Holden, K., & Butler, J. (2003). *Universal principles of design*. Gloucester, Mass: Rockport.

Link, C. (2009). Patterns for the commercial use of Open Source. In A. Kelly, & M. Weiss (Eds.), *EuroPLoP 2009. Proceedings of the 14th European Conference on Pattern Languages of Programs.* Irsee, Bavaria.

Lloyd, D. E. (2004). *Radiant cool: a novel theory of consciousness*. Cambridge, Mass: MIT Press.

Löbner, S. (2002). *Understanding semantics*. Understanding language series. London: Arnold.

Lopes, R., & Carriço, L. (2007). Patterns for Time-based Hypermedia Artifacts. In U. Zdun, & L. Hvatum (Eds.). *EuroPLoP 2006. Proceedings of the 11th European Conference on Pattern Languages of Programs.* (pp.247-276). Konstanz: Universitätsverlag Konstanz.

Lyardet, F., & Rossi, G. (1998). Patterns for Dynamic Websites. In *Proceedings of the 5th Conference on Pattern Languages of Programs*. http://hillside.net/plop/plop98/final_submissions/P56.pdf.

Lyardet, F., Rossi, G., & Schwabe, D. (1998). Using Design Patterns in Educational Multimedia applications. *AACE EDMEDIA Conference*. Freiburg, Germany.

Lyardet, F., Rossi, G., & Schwabe, D. (2000). Patterns for Adding Search Capabilities to Web Information Systems. In P. Dyson, & M. Devos (Eds.), *EuroPLoP 1999. Proceedings of the 4th European Conference on Pattern Languages of Programs*. (pp.189-202). Konstanz: Universitätsverlag Konstanz.

Mader, S. (2008). *Wikipatterns*. Indianapolis, IN: Wiley Pub.

Mahemoff, M. (2006). *Ajax design patterns*. Sebastopol, CA: O'Reilly.

Mainzer, K. (2008). *Komplexität*. Paderborn: UTB.

Majetschak, S. (2007). *Ästhetik zur Einführung*. Hamburg: Junius.

Malone, E., & Crumlish, C. (2009). *Designing social interfaces*. Sebastopol, CA: O'Reilly Media.

Marquardt, K. (2005). Platonic Schizophrenia. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp.85-108). Konstanz: Universitätsverlag Konstanz.

Marshall, S. P. (1995). *Schemas in problem solving*. Cambridge: Cambridge University Press.

Mayer, R. E. (2001). *Multimedia Learning*. Cambridge: Cambridge University Press

Meszaros, G., & Doble, J. (1997). A pattern language for pattern writing. In R.C. Martin, D. Riehle, & F. Buschmann (Eds.), *Pattern Languages of Program Design* (pp. 529-574). Boston, MA: Addison-Wesley Longman Publishing.

Metzinger, T. (Ed.) (2000). *Neural Correlates of Consciousness*. Cambridge, MA: MIT Press

Minsky, M. (1977). *Frame-system theory. Thinking: Readings in Cognitive Science*. Cambridge: Cambridge University Press

Mislevy, R.J., Chudowsky, N., Draney, K., Fried, R., Gaffney, T., Haertel, G., Hafter, A., Hamel, L., Kennedy, C., Long, K., Morrison, A.L., Murphy, R., Pena, P., Quellmalz, E., Rosenquist, A., Songer, N.B., Schank, P., Wenk, A., & Wilson, M.R. (2003). Design Patterns for Assessing Science Inquiry. *PADI Technical Report 1*. SRI International, Menlo Park, CA.

Mitchell, M. (2009). *Complexity: A guided tour*. Oxford: Oxford University Press.

Mitchell, M. T. (2006). *Michael Polanyi: The art of knowing*. Wilmington, Del: ISI Books.

Mor, Y. (2010). *A Design Approach to Research in Technology Enhanced Mathematics Education*. Doctoral thesis. http://eprints.ioe.ac.uk/6478/1/mor2010.pdf

Mor, Y., & Winters, N. (2007): Design approaches in technology enhanced learning. *Interactive Learning Environment.*, 15(1). 61-75.

Mor, Y., & Winters, N. (2008). A language of patterns for mathematical learning. In T. Schümmer, & A. Kelly (Eds.), *EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs*. Irsee, Bavaria.

Moreland, J. P. (2001). *Universals*. Montreal: McGill-Queen's University Press.

Moreno, R., & Mayer, R.E.. (1999) Cognitive Principles of Multimedia Learning: The Role of Modality and Contiguity. *Journal of Educational Psychology*. 91. 358-368.

Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. In *Proceedings of the first international workshop on Computing education research (ICER '05)* (pp. 57-67). New York, NY: ACM.

Naur, P. (1985). Programming as Theory Building. *Micorprocessing and Microprogramming*. 15.

Neumann, S., Derntl, M., & Oberhuemer, P. (2011). The Essential Structure of Teaching Method Descriptions. In C. Kohls, & J. Wedekind, J. (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions.* Hershey: Information Science Pub.

Neuweg, G. H. (2001). *Könnerschaft und implizites Wissen: Zur lehr-lerntheoretischen Bedeutung der Erkenntnis- und Wissenstheorie Michael Polanyis*. Internationale Hochschulschriften, Bd. 311. Münster: Waxmann.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice-Hall.

Noble, J. (1998). Classifying relationships between object-oriented design patters. *Australian Software Engineering Conference (ASWEC)*. 98-107.

Noble, J., & Biddle, R. (2002). Patterns as Signs. In Magnusson (Ed.), ECOOP'02 – Object-Oriented Programming. Málaga, Spain, July 2002. *Lecture Notes in Computer Science*. 2374. Berlin: Springer.

Noble, J., Biddle, R., & Tempero, E. (2006). Patterns as Signs: A Semiotics of Object-Oriented Design Patterns. *An International Journal on Communication, Information Technology and Work*. 2 (1). 3-40.

Ohlsson, S. (1984). Restructuring revisited, II: An information processing theory of restructuring and insight. *Scandinavian Journal of Psychology*. 25. 117-129

Olson, D. (1995). *Hands In View*. http://c2.com/cgi/wiki?HandsInView (01.08.2011)

Olson, D. (1998). Patterns on the Fly. In L. Rising, (Ed.) *The Pattern Handbook* (pp.141-170). Cambridge: Cambridge University Press.

Oram, A., and Wilson, G. (2007). *Beautiful code*. North Sebastapol, Calif: O'Reilly.

Pane, J.F., Ratanamahatana, C.A., & Myers, B.A. (2001). Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems. *International Journal of Human-Computer Studies*. 54(2). 237-264.

Pedagogical Pattern Project (2007). http://www.pedagogicalpatterns.org/

Piaget, J. (1952). *The Origins of Intelligence in Children*. New York: International University Press.

Piaget, J. (1971). *Biology and Knowledge*. Edinburgh: Edinburgh University Press.

Piaget, J. (1975). *Der Aufbau der Wirklichkeit beim Kinde*. Stuttgart: Ernst Klett.

Piaget, J., Fatke, R., and Kober., H. (2003). *Meine Theorie der geistigen Entwicklung*. Beltz Taschenbuch, 142. Weinheim: Beltz Verlag.

Piaget, J., & Inhelder, B. (1969). *The psychology of the child*. New York: Basic Books.

Polanyi, M. (1958). *Personal Knowledge: Towards a Post-Critical Philosophy*. Chicago: University Of Chicago Press.

Popper, K. R. (1972). *The logic of scientific discovery*. London: Hutchinson.

Quibeldey-Cirkel, K. (1999a) *Entwurfsmuster – Design Patterns in der objektorientierten Softwaretechnik*. Berlin: Springer.

Quibeldey-Cirkel, K. (1999b). ETHOS: A Pedagogical Pattern. In J. Coldewey, & P. Dyson (Eds), *Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing* (pp. 261-268). Konstanz: Universitätsverlag Konstanz.

Quilici, J.L., and Mayer, R.E. (1996). Role of examples in how students learn to categorize statistics word problems. *Journal of Educational Psychology*. 88. 144-161.

Quillien, J., (2008). *Delight's muse on Christopher Alexander's The nature of order: A summary and personal interpretation*. Ames, Iowa: Culicidae Architectural Press.

Quillien, J., Rostal, P., West, D. (2009). Agile, Anthropology, and Alexander's Architecture: An Essay in Three Voices. *OOPSLA 2009*. Orlando, FL, USA.

Raveh, A. (2005). Patterns for Interface Design. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp.123-132). Konstanz: Universitätsverlag Konstanz.

Raveh, A., & Homsky, O. (2008). Pattern Language for Online Communities. In L. Hvatum, & T. Schümmer (Eds.), *EuroPLoP 2007. Proceedings of the 12th European Conference on Pattern Languages of Programs*. (pp.121-148). Konstanz: Universitätsverlag Konstanz.

Reißing, R. (1999). A Presentation Pattern Language. In J. Coldewey, & P.Dyson (Eds.), *EuroPLoP 1998. Proceedings of the 3rd European Conference on Pattern Languages of Programs* (pp. 269-288). Konstanz: Universitätsverlag Konstanz.

Remy, C., Weiss, M., Ziefle, M., & Borchers, J. (2010). A Pattern Language for Interactive Tabletops in Collaborative Workspaces. In M. Weiss, & P. Avgeriou (Eds.), *EuroPLoP 2010. Proceedings of the 15th European Conference on Pattern Languages of Programs*.

Retalis, S., Georgiakakis, P., & Dimitriadis, Y. (2006). Eliciting design patterns for e-learning systems. *Computer Science Education*. 16(2). 105–118.

Richards, B.H. (1998). Frameworks and Design Patterns. In L. Rising (Ed.). *The Pattern Handbook* (pp.375-382).Cambridge: Cambridge University Press.

Rieber, L.P. (1990). *Computers, graphics, & learning*. Englewood Cliffs: Prentice Hall.

Rieber, L., & Kini, A.S. (1991). Theoretical foundations of instructional applications of computer-generated animated visuals. *Journal of Computer-Based Instruction* (pp. 83-88). 18.

Riehle, D., & Züllighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems*. 2 (1). 3-13.

Rising, L. (1997). Customer Interaction Patterns. In *Proceedings of the 4th Conference on Pattern Languages of Programs*. http://hillside.net/plop/plop97/Proceedings/rising.pdf (01.08.2011).

Rising, L. (1998). *The Pattern Handbook*. Cambridge: Cambridge University Press

Rising, L. (2007). Understanding the Power of Abstraction in Patterns. *IEEE Software*. July/August 2007.

Rising, L., & Manns, M. L. (2005). *Fearless change: Patterns for introducing new ideas*. Boston: Addison-Wesley.

Rising, L., & Rehmer, K. (2010). Patterns for sustainable development. *PLoP '09 Proceedings of the 16th Conference on Pattern Languages of Programs*. Reno, Nevada.

Rumelhart, D. E., & Norman, D.A. (1978). Accretion, tuning, and restructuring: Three models of learning. In J.W. Cotton, and R. Klatzky (Eds.). Semantic Factors in Cognition. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rumelhart, D. E., & Ortony, A. (1977). The representation of knowledge in memory. In R. C. Anderson, J. R. Spiro, & W. E. Montague (Eds), *Schooling and the acquisition of knowledge*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Salah, D., & Zeid, A. (2009). PLITS:A Pattern Language for Intelligent Tutoring Systems. In A. Kelly, & M. Weiss (Eds.), *EuroPLoP 2009. Proceedings of the 14th European Conference on Pattern Languages of Programs.* Irsee, Bavaria.

Saunders, D. (1998). Patterns: The New Building Blocks for Reusable Software Architectures. In L. Rising (Ed.), *The Pattern Handbook* (pp.45-47). Cambridge: Cambridge University Press.

Schadewitz, N. (2008). Design Patterns for Cross-Cultural Computer Supported Design. In L. Hvatum,
& T. Schümmer (Eds.), *EuroPLoP 2007. Proceedings of the 12th European Conference on Pattern Languages of Programs* (pp. 409-428). Konstanz: Universitätsverlag Konstanz.

Schadewitz, N., & Jachna, T. (2007). Comparing inductive and deductive methodologies for design patterns identification and articulation. *International Design Research Conference*. 12 (15).

Schank, R. C. (1975). The structure of episodes in memory. In D. Bobrow, & A. Collins (Eds.), *Representation and understanding* (pp. 237-272). New York: Academic Press.

Schank, R. C., & Abelson, R. (1977). *Scripts, plans, goals and understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Scharlau, I. (2007). *Jean Piaget zur Einführung*. Hamburg: Junius.

Schiffer, S. (1998). *Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten.* Bonn: Addison-Wesley.

Schmettow, M. (2007). User Interaction Design Patterns for Information Retrieval Systems. In U. Zdun, & L. Hvatum (Eds.), *EuroPLoP 2006. Proceedings of the 11th European Conference on Pattern Languages of Programs* (pp.489-512). Konstanz: Universitätsverlag Konstanz.

Schmolitzky, A. (2008). Patterns for Teaching Software in Classroom. In T. Schümmer, & A. Kelly (Eds.), *EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs*. Irsee, Bavaria.

Schmolitzky, A., & Schümmer, T. (2008). Patterns for Supervising Thesis Projects. In T. Schümmer, & A. Kelly, (Eds.), *EuroPLoP 2008. Proceedings of the 13th European Conference on Pattern Languages of Programs*. Irsee, Bavaria.

Schmolitzky, A., & Schümmer, T. (2011). Towards Pedagogical Patterns on Feedback. In C. Kohls, & J. Wedekind (Eds). *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions*. Hershey: Information Science Pub.

Schnädelbach, H. (2002). *Erkenntnistheorie zur Einführung*. Hamburg: Junius.

Schnelle, D., & Lyardet, F. (2007). Voice User Interface Design Patterns. In U. Zdun, & L. Hvatum (Eds.). *EuroPLoP 2006. Proceedings of the 11th European Conference on Pattern Languages of Programs*. Konstanz: Universitätsverlag Konstanz.

Schnelle, D., Lyardet, F., & Wei, T. (2006). Audio Navigation Patterns. In A. Longshaw, & U. Zdun (Eds.). *EuroPLoP 2005. Proceedings of the 11th European Conference on Pattern Languages of Programs* (pp.237-260) Konstanz: Universitätsverlag Konstanz.

Schnelle-Walka, D. (2010). A Pattern Language for Error Management in Voice User Interfaces. In M. Weiss, & P. Avgeriou (Eds.), *EuroPLoP 2010. Proceedings of the 15th European Conference on Pattern Languages of Programs*. Irsee, Bavaria.

Schnotz, W. (1994). *Aufbau von Wissenstrukturen*. Weinheim:Beltz.

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., & Buschmann, F. (2006). *Security patterns: Integrating security and systems engineering*. Chichester, England: John Wiley & Sons.

Schumann, H., & Müller, W. (2005). *Visualisierung. Grundlagen und allgemeine Methoden*. Berlin: Springer.

Schümmer, T. (2005a). *A Pattern Approach for End-User Centered Groupware Development*. Köln: Josef Eul Verlag.

Schümmer, T. (2005b). Patterns for Building Communities in Collaborative Systems. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp.379-440). Konstanz: Universitätsverlag Konstanz.

Schümmer, T., & Lukosch, S. (2007). *Patterns for computer-mediated interaction*. Wiley series in software design patterns. Chichester, England: John Wiley & Sons.

Schümmer, T., & Tandler, P. (2008). Patterns for Technology Enhanced Meetings. In L. Hvatum, & T. Schümmer (Eds.), *EuroPLoP 2007. Proceedings of the 12th European Conference on Pattern Languages of Programs* (pp.97-120). Konstanz: Universitätsverlag Konstanz.

Scott, B., & Neil, T. (2008). *Designing web interfaces: Principles and patterns for rich interactions*. Sebastopol, CA: O'Reilly.

Shannon, C.E., & Weaver, W. (1949). *The mathematical theory of information*. Urbana: University of Illinois Press.

Simon, H.A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society.* 74. 29-39.

Simon, H.A. (1973). Does Scientific Discovery Have A Logic*? Philosophy of Science*. 40. 471- 480.

Simon, H. A. (1996). *The science of the artificial*. 3rd edition. Cambridge: M.I.T. Press.

Steindl, C. (2001). Pedagogical Pattern: Self Test. In M. Devos, & A. Rüping (Eds.). *EuroPLoP 2000. Proceedings of the 4th European Conference on Pattern Languages of Programs* (pp. 167-172). Konstanz: Universitätsverlag Konstanz.

Stewart, I. (2007). *Why beauty is truth: A history of symmetry*. New York: Basic Books.

Stewart, I., & Golubitsky, M. (1992). *Fearful symmetry: Is God a geometer?* London: Penguin.

Strauss, A. L., & Corbin, J. M. (1990). *Basics of qualitative research: Grounded theory procedures and techniques*. Newbury Park, Calif: Sage Publications.

Sweller, J., van Merriënboer, J. J. G., & Paas, F. G. W. C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review.* 10. 251-296.

Taleb, N. (2007). *The black swan: The impact of the highly improbable*. New York: Random House.

Thompson, D. A. W. (1942). *On growth and form*. Cambridge: Cambridge University Press.

Tidwell, J. (1998). Interaction Patterns. In *Proceedings of the 5th Conference on Pattern Languages of Programs*. http://hillside.net/plop/plop98/final_submissions/P29.pdf (03.08.2011).

Tidwell, J. (2005). *Designing Interfaces*. Sebastopol: O'Reilly Media.

Tufte, E.R. (1990). *Envisioning Information*. Cheshire: Graphics Press.

Van Duynie, D., Landay, J. A., & Hong, J. I. (2004). *The Design of Sites*. Boston: Addison-Wesley.

VanLehn, K. (1989). Problem Solving and Cognitive Skills Acquisition. In M.I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 527-579). Cambridge: MIT Press.

Van der Veer, G. C., & Melguizo, M. C. (2002). Mental Models. In J.A. Jacok, & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, evolving Technologies and emerging Applications* (pp. 52-80). New York, NY: Lawrence Erlbaum & Associates.

Velázquez, K. (2012) A Pattern for Learning Phraseology of a Foreign Language. *18th European Pattern Languages of Programs conference.* Writers' Workshop version. Irsee, Bavaria.

Villasclaras-Fernández, E.D., Hernández-Leo, D., Asensio-Pérez, J.I, Dimitriadis, Y., & Martínez-Monés, A. (2011). Linking CSCL Script Design Patterns: Connections between Assessment and Learning Patterns. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions*. Hershey: Information Science Pub.

Vlissides, J. (1996). Seven Habits of Successful Pattern Writers. *C++ Report*. November/December 1996.

Vlissides, J. (1997). Patterns: The Top Ten Misconceptions. *Object Magazine*. March 1997.

Vlissides, J. (1998). Pattern Hatching – Perspectives from the "Gang of Four". In L. Rising (Ed.), *The Pattern Handbook* (pp.505-513). Cambridge: Cambridge University Press.

Vogel, R., & Wippermann S. (2005). Dokumentation didaktischen Wissens in der Hochschule. Didaktische - Design Patterns als eine Form des Best-Practice-Sharing im Bereich von IKT in der Hochschullehre. In K. Fuchs-Kittowski, W. Umstätter, & R. Wagner-Döbler (Eds.), *Wissenschaftsforschung Jahrbuch 2004* (pp. 17-42). Berlin: Gesellschaft für Wissenschaftsforschung.

Vogiatzis, D., Tzanavari, A., Retalis, S., Avgeriou, P., & Papasalouros, A. (2005). The Learner's Mirror. Designing a User Modelling Component in Adaptive Hypermedia Educational Systems. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp. 629-644). Konstanz: Universitätsverlag Konstanz.

Volk, T. (1995). *Metapatterns across space, time, and mind*. New York: Columbia University Press.

Völter, M., & Fricke, A. (2001). SEMINARS. In M. Devos & A. Rüping (Eds.). *EuroPLoP 2000. Proceedings of the 4th European Conference on Pattern Languages of Programs* (pp.87-128). Konstanz: Universitätsverlag Konstanz.

Waguespack, L. J. (2010). *Thriving systems theory and metaphor-driven modeling*. London: Springer.

Wake, W. C. (1998). Patterns for Interactive Applications. In *Proceedings of the 5th Conference on Pattern Languages of Programs*. http://hillside.net/plop/plop98/final_submissions/P44.pdf (03.08.2011).

Ware, C. (2004). *Information Visualization – Perception for Design*. San Francisco: Morgan Kaufmann Publishers.

von Wartensleben, G. (1913). Über den Einfluss der Zwischenzeit auf die Reproduktion gelesener Buchstaben. *Zeitschrift für Psychologie und Physiologie der Sinnesorgane*. 64. 321-385.

Watzlawick, P. (1976). *How real is real? Confusion, disinformation, communication*. New York: Random House.

Watzlawick, P, Bavelas, J. B., & Jackson, D.D. (1967). *Pragmatics of Human Communication. A Study of Interactional Patterns, Pathologies, and Paradoxes*. New York: W.W. Norton & Company.

Wedekind, J. (2011). Patterns and Instructional Methods: A Practitioner's Approach. In C. Kohls, & J. Wedekind (Eds), *Investigations of E-Learning Patterns: Context Factors, Problems and Solutions*. Hershey: Information Science Pub.

Weinberger, A., Ertl, B., Fischer, F., & Mandl, H. (2005). Epistemic and social scripts in computer-supported collaborative learning. *Instructional Science*. 33. 1-30.

Weir, C., & Noble, J. (2004). A Window in your Pocket. Some Small Patterns for User Interfaces. In D. Schütz, & K. Marquardt (Eds.), *EuroPLoP 2003. Proceedings of the 9th European Conference on Pattern Languages of Programs* (pp. 19-36). Konstanz: Universitätsverlag Konstanz.

Weir, C., Noble, J., Martin, A., & Biddle, R. (2005): My Friend the Customer. In K. Henney, & D. Schütz (Eds.), *EuroPLoP 2004. Proceedings of the 8th European Conference on Pattern Languages of Programs* (pp. 199-214). Konstanz: Universitätsverlag Konstanz.

Wellhausen, T., & Fießer, A. (2011). How to write a pattern? A guideline for first-time pattern authors. *EuroPLoP 2011. Proceedings of the 18th European Conference on Pattern Languages of Programs.* Irsee, Bavaria.

Wertheimer, M. (1938). *Laws of organization in perceptual forms*. London: Harcourt, Brace, and Jovanovitch.

Westermann, R. (2000). *Wissenschaftstheorie und Experimentalmethodik: Ein Lehrbuch zur psychologischen Methodenlehre*. Göttingen u.a.: Hogrefe, Verl. für Psychologie.

Westermann, R., & Gerjets, P. (1994). Induktion. In T. Herrmann, & W. Tack (Eds.), *Enzyklopädie der Psychologie: Themenbereich B. Serie I, Bd. 1, Methodologische Grundlagen der Psychologie* (pp. 428-472). Göttingen: Hogrefe.

Whitehead, A. N., & Russell, B. (1925). *Principia mathematica*. Cambridge, England: The University Press

Whitworth, E., & Biddle, R. (2006). Share and Enjoy! Patterns for Successful Knowledge Sharing in Large Online Communities. In A. Longshaw, & U. Zdun (Eds.). *EuroPLoP 2005. Proceedings of the 15th European Conference on Pattern Languages of Programs* (pp. 1-20). Konstanz: Universitätsverlag Konstanz.

Winters, N., & Mor, Y. (2009). Dealing with abstraction: Case study generalisation as a method for eliciting design patterns. *Computers in Human Behavior*. 25 (5). 1079-1088.

Wippermann, S. (2008). *Didaktische Design Patterns: Zur Dokumentation und Systematisierung didaktischen Wissens und als Grundlage einer Community of Practice*. Saarbrücken: VDM Verl. Dr. Müller.

Wirfs-Brock, R. (2008). What Drives Design. *Companion To the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications.* http://portal.acm.org/citation.cfm?id=1449814.1501018. (05.05.2011) Nashville, TN: ACM.

Wirfs-Brock, R. (2010). The Nature of Order: Inspiration or Esoteric Distraction? *Keynote at PLoP 2010*. Reno, Nevada.

Wohner, W. (2003). *EOS: An Epistemological Ontology-driven System for Knowledge Processing*. Doctoral Dissertation. http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2003/wohner.pdf (accessed 1.8.2011).

Wundt, W. (1907). *Logik der exakten Wissenschaft. Eine Untersuchung der Prinzipien der Erkenntnis und der Methoden wissenschaftlicher Forschung. II. Band.* 3rd edition. Stuttgart: Verlag Ferdinand Enke.

Zehnpfennig, B. (1997). *Platon zur Einführung*. Hamburg: Junius.

Zhao, L. (2008). Patterns, Symmetry, and Symmetry Breaking. *Communications of the ACM*. 51(3).

Zhao, L., & Coplien, J.O. (2002). Symmetry in Class and Type Hierarchy. In J. Noble, & J. Potter (Eds), *40th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific 2002).* Sydney, Australia: Australian Computer Society, Inc.

Zhao, L., & Coplien, J.O. (2003). Understanding Symmetry in Object-Oriented Languages. *Journal of Object Technology*. 2(5). 123-134.

Zimbardo, P. G., Gerrig, R. J., & Graf, R. (2004). *Psychologie*. Pearson-Studium. München: Pearson.

Zimmermann, B. (2008): *Pattern-basierte Prozessbeschreibung und -unterstützung: Ein Werkzeug zur Unterstützung von Prozessen zur Anpassung von E-Learning-Materialien*. Doctoral Dissertation. http://tuprints.ulb.tu-darmstadt.de/1217/1/Dissertation_Birgit_Zimmermann.pdf (19.05.2011).

Appendix A – Data of experiments

Phase 1

The pair comparisons used the following graphics:

1 Night watchman (F) , Map - UK (E)
2 Map - UK (E) , Flower (A)
3 State of aggregation (K) , Vocabulary (H)
4 Night watchman (F) , Skeleton - dragable label (N)
5 Presentation Media (L) , Skeleton - dragable label (N)
6 Computer Board (D) , Team (M)
7 Skeleton - single labels / click (C) , Map - Scandinavia (J)
8 Vocabulary (H) , Map - UK (E)
9 Presentation Media (L) , Camera (B)
10 Vocabulary (H) , Skeleton - Clickable label (C)
11 State of aggregation (K) , Animals (G)
12 Vocabulary (H) , X-Ray (I)
13 Vocabulary (H) , Flower (A)
14 Computer Board (D) , Map - UK (E)
15 Vocabulary (H) , Computer Board (D)
16 Team (M) , Flower (A)
17 Vocabulary (H) , Map - Scandinavia (J)
18 Animals (G) , Camera (B)
19 Map - Scandinavia (J) , Map - UK (E)
20 Presentation Media (L) , Night watchman (F)
21 X-Ray (I) , Map - UK (E)
22 Animals (G) , Map - Scandinavia (J)
23 X-Ray (I) , Skeleton - dragable label (N)
24 Animals (G) , "Plantine"
25 Map - Scandinavia (J) , Skeleton - dragable label (N)
26 Animals (G) , Presentation Media (L)
27 Vocabulary (H) , Animals (G)
28 State of aggregation (K) , Night watchman (F)
29 Animals (G) , Night watchman (F)
30 Night watchman (F) , Camera (B)
31 Animals (G) , Skeleton - single labels / click (C)
32 Vocabulary (H) , Presentation Media (L)
33 Vocabulary (H) , Skeleton - dragable label (N)
34 Presentation Media (L) , Skeleton - single labels / click (C)
35 Flower (A) , Skeleton - dragable label (N)
36 Animals (G) , Flower (A)
37 Skeleton - single labels / click (C) , Computer Board (D)
38 Camera (B) , Skeleton - dragable label (N)
39 State of aggregation (K) , Skeleton - single labels / click (C)
40 Skeleton - single labels / click (C) , Team (M)
41 X-Ray (I) , Flower (A)
42 Skeleton - single labels / click (C) , Map - UK (E)
43 State of aggregation (K) , Presentation Media (L)
44 Map - Scandinavia (J) , X-Ray (I)
45 Map - Scandinavia (J) , Team (M)
46 Night watchman (F) , Flower (A)
47 X-Ray (I) , Team (M)
48 Night watchman (F) , Skeleton - single labels / click (C)
49 Night watchman (F) , Map - Scandinavia (J)
50 Presentation Media (L) , Computer Board (D)
51 State of aggregation (K) , Map - Scandinavia (J)
52 Map - Scandinavia (J) , Camera (B)
53Vocabulary (H) , Camera (B)
54 State of aggregation (K) , Map - UK (E)
55 Camera (B) , Map - UK (E)
56 Presentation Media (L) , Flower (A)
57 Skeleton - single labels / click (C) , Skeleton - dragable label (N)
58 State of aggregation (K) , Camera (B)
59 Presentation Media (L) , X-Ray (I)
60 Skeleton - single labels / click (C) , Camera (B)
61 State of aggregation (K) , "Rhötgen"
62 State of aggregation (K) , Computer Board (D)
63 State of aggregation (K) , Flower (A)
64 Vocabulary (H) , Night watchman (F)
65 Skeleton - single labels / click (C) , X-Ray (I)
66 Animals (G) , X-Ray (I)
67 Team (M) , Camera (B)
68 Map - Scandinavia (J) , Computer Board (D)
69 X-Ray (I) , Computer Board (D)
70 Team (M) , Map - UK (E)
71Computer Board (D) , Camera (B)
72 Presentation Media (L) , Team (M)
73 Animals (G) , Map - UK (E)
74 Night watchman (F) , Team (M)
75 State of aggregation (K) , Team (M)
76 Vocabulary (H) , Team (M)
77 Map - Scandinavia (J) , Flower (A)
78 Night watchman (F) , X-Ray (I)
79 Skeleton - single labels / click (C) , Flower (A)
80 State of aggregation (K) , Skeleton - dragable label (N)
81 Computer Board (D) , Skeleton - dragable label (N)
82 Presentation Media (L) , Map - UK (E)
83 Map - UK (E) , Skeleton - dragable label (N)
84 Camera (B) , Flower (A)
85 Night watchman (F) , Computer Board (D)
86 Animals (G) , Team (M)
87 Presentation Media (L) , Map - Scandinavia (J)
88 Computer Board (D) , Flower (A)
89 Team (M) , Skeleton - dragable label (N)
90 X-Ray (I) , Camera (B)
91 Animals (G) , Skeleton - dragable label (N)
92 State of aggregation (K) , Vocabulary (H)
93 Skeleton - single labels / click (C) , Map - Scandinavia (J)
94 Vocabulary (H) , Skeleton - single labels / click (C)
95 Computer Board (D) , Team (M)
96 Vocabulary (H) , X-Ray (I)
97 Map - UK (E) , Flower (A)
98 Night watchman (F) , Skeleton - dragable label (N)
99 State of aggregation (K) , Animals (G)
100 Presentation Media (L) , Skeleton - dragable label (N)
101 Presentation Media (L) , Camera (B)
102 Night watchman (F) , Map - UK (E)
103 Vocabulary (H) , Map - UK (E)

H1.1, H1.2, H1.3 - This table shows the outcome of pair comparisons for all groups. f means the actual mean is outside the expected range. n.s. means the actual mean is within the expected range but the uppler/lower limit of the confidence interval (p=0,05) is outside the expected range (indicated in red). * means the upper and lower limit of the confidence interval (p=0,05) is within the expected range (indicated in green) but the upper/lower limit of the confidence interval (p=0,01) is without the expected range. ** means that both upper and lower limit of the confidence interval (p=0,01) are within the expected range.

#	Interaction type	Expected outcome for mean	Actual outcome for mean (all groups)	Standard deviation	Confidence interval (p=0,05) from/to		Confidence interval from/to		Significance
1	Different class	0-33%	23,94%	0,426739645	14,61%	35,54%	12,33%	39,17%	n.s
2	Same class	68-100%	28,17%	0,449823082	18,13%	40,10%	15,59%	43,77%	f
3	Super/ /Sub class	33-68%	56,34%	0,495966672	44,05%	68,09%	40,48%	71,33%	n.s
4	Super/Sub class	33-68%	69,01%	0,462435357	56,92%	79,46%	53,25%	82,16%	f
5	Same class	68-100%	53,52%	0,498758626	41,29%	65,45%	37,77%	68,78%	f
6	Different class	0-33%	53,52%	0,498758626	41,29%	65,45%	37,77%	68,78%	f
7	Different class	0-33%	19,72%	0,397871743	11,22%	30,87%	9,23%	34,42%	*
8	Different class	0-33%	38,03%	0,485455998	26,76%	50,33%	23,73%	53,97%	f
9	Different class	0-33%	5,63%	0,230573318	1,56%	13,80%	0,96%	16,69%	**
10	Different class	0-33%	26,76%	0,442710807	16,94%	38,59%	14,48%	42,25%	n.s
11	Super/Sub class	33-68%	54,93%	0,497563992	42,66%	66,77%	39,12%	70,06%	n.s
12	Same class	68-100%	81,69%	0,38674733	70,73%	89,87%	67,21%	91,76%	*
13	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
14	Same class	68-100%	91,55%	0,278146728	82,51%	96,84%	79,41%	97,78%	**
15	Different class	0-33%	22,54%	0,417813999	13,46%	34,00%	11,27%	37,61%	n.s
16	Different class	0-33%	16,90%	0,374764358	9,05%	27,66%	7,28%	31,14%	**
17	Different class	0-33%	77,46%	0,417813999	66,00%	86,54%	62,39%	88,73%	f
18	Different class	0-33%	2,82%	0,165455495	0,34%	9,81%	0,15%	12,41%	**
19	Different class	0-33%	28,17%	0,449823082	18,13%	40,10%	15,59%	43,77%	n.s
20	Super/Sub class	33-68%	59,15%	0,491547278	46,84%	70,68%	43,23%	73,82%	n.s
21	Different class	0-33%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	**
22	Different class	0-33%	80,28%	0,397871743	69,13%	88,78%	65,58%	90,77%	f
23	Super/Sub class	33-68%	66,20%	0,473038187	53,99%	77,00%	50,32%	79,84%	n.s
24	Different class	0-33%	26,76%	0,442710807	16,94%	38,59%	14,48%	42,25%	n.s
25	Same class	68-100%	94,37%	0,230573318	86,20%	98,44%	83,31%	99,04%	**
26	Super/Sub class	33-68%	45,07%	0,497563992	33,23%	57,34%	29,94%	60,88%	n.s
27	Same class	68-100%	91,55%	0,278146728	82,51%	96,84%	79,41%	97,78%	**
28	Super/Sub class	33-68%	43,66%	0,495966672	31,91%	55,95%	28,67%	59,52%	n.s
29	Same class	68-100%	81,69%	0,38674733	70,73%	89,87%	67,21%	91,76%	*
30	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
31	Different class	0-33%	8,45%	0,278146728	3,16%	17,49%	2,22%	20,59%	**
32	Super/Sub class	33-68%	49,30%	0,499950404	37,22%	61,44%	33,80%	64,88%	**
33	Super/Sub class	33-68%	70,42%	0,45639019	58,41%	80,67%	54,73%	83,29%	f
34	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
35	Different class	0-33%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	**
36	Different class	0-33%	12,68%	0,332704558	5,96%	22,70%	4,57%	26,03%	**
37	Same class	68-100%	30,99%	0,462435357	20,54%	43,08%	17,84%	46,75%	f
38	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
39	Different class	0-33%	4,23%	0,201166998	0,88%	11,86%	0,48%	14,62%	**
40	Different class	0-33%	18,31%	0,38674733	10,13%	29,27%	8,24%	32,79%	**
41	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
42	Same class	68-100%	22,54%	0,417813999	13,46%	34,00%	11,27%	37,61%	f
43	Same class	68-100%	91,55%	0,278146728	82,51%	96,84%	79,41%	97,78%	**
44	Super/Sub class	33-68%	63,38%	0,481764264	51,10%	74,50%	47,44%	77,47%	n.s
45	Same class	68-100%	91,55%	0,278146728	82,51%	96,84%	79,41%	97,78%	**
46	Different class	0-33%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	**
47	Super/Sub class	33-68%	57,75%	0,493962758	45,44%	69,39%	41,85%	72,58%	n.s
48	Different class	0-33%	4,23%	0,201166998	0,88%	11,86%	0,48%	14,62%	**
#	**Interaction type**	**Expected**	**Actual**	**Stan. deviation**	**Conf. interval (0,05)**		**Conf. interval (0,01)**		**Significance**
49	Super/Sub class	33-68%	60,56%	0,488714129	48,25%	71,97%	44,62%	75,05%	n.s

50	Different class	0-33%	7,04%	0,255857776	2,33%	15,67%	1,55%	18,68%	**
51	Same class	68-100%	81,69%	0,38674733	70,73%	89,87%	67,21%	91,76%	*
52	Different class	0-33%	2,82%	0,165455495	0,34%	9,81%	0,15%	12,41%	**
53	Different class	0-33%	5,63%	0,230573318	1,56%	13,80%	0,96%	16,69%	**
54	Different class	0-33%	5,63%	0,230573318	1,56%	13,80%	0,96%	16,69%	**
55	Same class	68-100%	7,04%	0,255857776	2,33%	15,67%	1,55%	18,68%	f
56	Different class	0-33%	5,63%	0,230573318	1,56%	13,80%	0,96%	16,69%	**
57	Different class	0-33%	21,13%	0,408207796	12,33%	32,44%	10,24%	36,02%	*
58	Different class	0-33%	1,41%	0,11783944	0,04%	7,60%	0,01%	10,00%	**
59	Super/Sub class	33-68%	59,15%	0,491547278	46,84%	70,68%	43,23%	73,82%	n.s
60	Same class	68-100%	84,51%	0,361837537	73,97%	92,00%	70,53%	93,65%	**
61	Super/Sub class	33-68%	49,30%	0,499950404	37,22%	61,44%	33,80%	64,88%	**
62	Different class	0-33%	1,41%	0,11783944	0,04%	7,60%	0,01%	10,00%	**
63	Different class	0-33%	8,45%	0,278146728	3,16%	17,49%	2,22%	20,59%	**
64	Same class	68-100%	81,69%	0,38674733	70,73%	89,87%	67,21%	91,76%	*
65	Different class	0-33%	12,68%	0,332704558	5,96%	22,70%	4,57%	26,03%	**
66	Same class	68-100%	83,10%	0,374764358	72,34%	90,95%	68,86%	92,72%	**
67	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
68	Different class	0-33%	19,72%	0,397871743	11,22%	30,87%	9,23%	34,42%	*
69	Different class	0-33%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	**
70	Different class	0-33%	14,08%	0,347861663	6,97%	24,38%	5,44%	27,77%	**
71	Same class	68-100%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	f
72	Same class	68-100%	74,65%	0,435026626	62,92%	84,23%	59,28%	86,60%	n.s
73	Different class	0-33%	14,08%	0,347861663	6,97%	24,38%	5,44%	27,77%	**
74	Super/Sub class	33-68%	64,79%	0,477628872	52,54%	75,76%	48,88%	78,66%	n.s
75	Same class	68-100%	73,24%	0,442710807	61,41%	83,06%	57,75%	85,52%	n.s
76	Super/Sub class	33-68%	64,79%	0,477628872	52,54%	75,76%	48,88%	78,66%	n.s
77	Different class	0-33%	7,04%	0,255857776	2,33%	15,67%	1,55%	18,68%	**
78	Same class	68-100%	90,14%	0,298112824	80,74%	95,94%	77,55%	97,05%	**
79	Same class	68-100%	63,38%	0,481764264	51,10%	74,50%	47,44%	77,47%	f
80	Same class	68-100%	80,28%	0,397871743	69,13%	88,78%	65,58%	90,77%	*
81	Different class	0-33%	25,35%	0,435026626	15,77%	37,08%	13,40%	40,72%	n.s
82	Different class	0-33%	7,04%	0,255857776	2,33%	15,67%	1,55%	18,68%	**
83	Different class	0-33%	15,49%	0,361837537	8,00%	26,03%	6,35%	29,47%	**
84	Same class	68-100%	84,51%	0,361837537	73,97%	92,00%	70,53%	93,65%	**
85	Different class	0-33%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	**
86	Super/Sub class	33-68%	73,24%	0,442710807	61,41%	83,06%	57,75%	85,52%	f
87	Same class	68-100%	81,69%	0,38674733	70,73%	89,87%	67,21%	91,76%	*
88	Same class	68-100%	11,27%	0,316196399	4,99%	21,00%	3,74%	24,26%	f
89	Same class	68-100%	100,00%	0	100,00%	100,00%	92,81%	100,00%	**
90	Different class	0-33%	4,23%	0,201166998	0,88%	11,86%	0,48%	14,62%	**
91	Super/Sub class	33-68%	63,38%	0,481764264	51,10%	74,50%	47,44%	77,47%	n.s
92/3r	Super/Sub class	33-68%	49,30%	0,499950404	37,22%	61,44%	33,80%	64,88%	**
93/7r	Different class	0-33%	7,04%	0,255857776	2,33%	15,67%	1,55%	18,68%	**
94/10r	Different class	0-33%	9,86%	0,298112824	4,06%	19,26%	2,95%	22,45%	**
95/6r	Different class	0-33%	22,54%	0,417813999	13,46%	34,00%	11,27%	37,61%	n.s
96/12r	Same class	68-100%	87,32%	0,332704558	77,30%	94,04%	73,97%	95,43%	**
97/2r	Same class	68-100%	12,68%	0,332704558	5,96%	22,70%	4,57%	26,03%	f
98/4r	Super/Sub class	33-68%	59,15%	0,491547278	46,84%	70,68%	43,23%	73,82%	n.s
99/11r	Super/Sub class	33-68%	45,07%	0,497563992	33,23%	57,34%	29,94%	60,88%	n.s
100/5r	Same class	68-100%	78,87%	0,408207796	67,56%	87,67%	63,98%	89,76%	n.s
101/9r	Different class	0-33%	1,41%	0,11783944	0,04%	7,60%	0,01%	10,00%	**
102/1r	Different class	0-33%	8,45%	0,278146728	3,16%	17,49%	2,22%	20,59%	**
103/8r	Different class	0-33%	15,49%	0,361837537	8,00%	26,03%	6,35%	29,47%	**

Phase 2

H2.3 - The table shows for each participant (id) the coherence performance after 1..12 pair comparisons. For example, participant “Seife” had 72,7% coherent judgements after 11 comparisons and 75% coherent judgement after 12 pair comparisons. The mean, SD and confidence intervals are given for the coherence performance of all participants after n pair comparions. For example, after 12 pair comparisons the mean is 65,84%. That is the average performance of all participants after 12 pair comparions. Based on the standard deviation we can calculate the confidence intervals (for alpha=0,05).

ids	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
Seife	0	0,5	0,667	0,5	0,4	0,5	0,571	0,625	0,667	0,7	0,727	0,75
Apfel	1	1	1	1	1	0,833	0,857	0,875	0,889	0,9	0,909	0,917
Hamburg	1	0,5	0,333	0,25	0,2	0,167	0,286	0,25	0,333	0,3	0,364	0,333
Baum	0	0	0,333	0,5	0,6	0,667	0,571	0,5	0,556	0,6	0,636	0,583
Wolke	0	0	0,333	0,5	0,4	0,333	0,429	0,375	0,444	0,4	0,455	0,5
Birne	1	1	1	1	0,8	0,833	0,857	0,875	0,889	0,9	0,818	0,833
Sonne	1	1	1	1	1	1	1	0,875	0,889	0,8	0,818	0,833
Kerze	1	1	1	1	1	1	0,857	0,75	0,778	0,7	0,636	0,583
Licht	1	1	0,667	0,5	0,4	0,5	0,571	0,625	0,667	0,7	0,636	0,667
Kiel	1	0,5	0,667	0,5	0,6	0,667	0,571	0,5	0,556	0,5	0,455	0,5
Banane	1	0,5	0,667	0,75	0,8	0,667	0,714	0,75	0,778	0,7	0,727	0,75
Eule	1	1	1	0,75	0,6	0,5	0,571	0,5	0,556	0,5	0,546	0,583
Lampe	0	0	0	0	0,2	0,167	0,286	0,375	0,444	0,5	0,546	0,5
Feuer	1	0,5	0,667	0,75	0,6	0,5	0,571	0,5	0,556	0,5	0,546	0,583
Katze	1	1	0,667	0,5	0,4	0,5	0,571	0,625	0,667	0,7	0,727	0,75
Blume	1	1	0,667	0,5	0,4	0,333	0,429	0,5	0,556	0,6	0,636	0,667
Regen	1	1	1	1	1	1	1	1	1	1	1	1
Hund	1	1	1	1	1	1	1	0,875	0,889	0,9	0,909	0,917
Wasser	0	0,5	0,667	0,5	0,6	0,667	0,714	0,75	0,778	0,8	0,727	0,75
Tiere	1	1	0,667	0,75	0,8	0,667	0,714	0,75	0,778	0,7	0,727	0,667
Koffer	1	0,5	0,667	0,75	0,6	0,5	0,571	0,625	0,667	0,6	0,546	0,583
Igel	1	1	1	1	0,8	0,833	0,857	0,875	0,889	0,9	0,818	0,833
Berlin	1	0,5	0,333	0,25	0,4	0,5	0,571	0,625	0,667	0,6	0,546	0,583
Bus	1	1	1	1	1	1	1	0,875	0,889	0,8	0,818	0,833
Bremen	1	0,5	0,667	0,75	0,6	0,5	0,571	0,625	0,667	0,6	0,636	0,667
Frosch	0	0,5	0,333	0,25	0,4	0,5	0,571	0,625	0,667	0,7	0,636	0,667
Elbe	1	0,5	0,667	0,5	0,4	0,333	0,429	0,5	0,444	0,5	0,455	0,417
Farbe	1	1	0,667	0,5	0,4	0,5	0,571	0,625	0,667	0,7	0,636	0,667
Hase	1	1	0,667	0,75	0,8	0,833	0,714	0,625	0,667	0,6	0,636	0,583
Genf	1	0,5	0,667	0,5	0,6	0,5	0,571	0,625	0,667	0,6	0,546	0,5
Haus	1	1	0,667	0,5	0,4	0,333	0,429	0,5	0,556	0,5	0,546	0,583
Eisen	1	0,5	0,667	0,75	0,6	0,667	0,714	0,75	0,667	0,7	0,727	0,75
Buch	1	1	1	1	1	1	1	1	1	0,9	0,909	0,917
Kran	1	1	1	1	1	1	1	1	1	0,9	0,909	0,917
Aula	0	0,5	0,667	0,5	0,6	0,667	0,571	0,625	0,667	0,7	0,727	0,75
Elch	1	1	0,667	0,5	0,4	0,333	0,286	0,375	0,333	0,3	0,364	0,333
Golf	1	0,5	0,667	0,5	0,4	0,333	0,286	0,25	0,333	0,4	0,455	0,417
Ampel	0	0,5	0,667	0,75	0,8	0,833	0,857	0,875	0,889	0,8	0,818	0,833
Biber	1	1	1	0,75	0,6	0,667	0,571	0,5	0,556	0,6	0,636	0,667
Acht	1	1	1	0,75	0,6	0,5	0,571	0,5	0,556	0,5	0,455	0,417
Indien	1	1	1	0,75	0,6	0,667	0,571	0,625	0,556	0,6	0,636	0,583
Post	1	0,5	0,333	0,25	0,4	0,5	0,571	0,625	0,667	0,7	0,727	0,75
Tulpe	1	1	0,667	0,75	0,6	0,667	0,714	0,75	0,667	0,7	0,727	0,667
Mais	1	1	0,667	0,5	0,6	0,667	0,714	0,75	0,778	0,8	0,818	0,833
Belgien	1	1	0,667	0,75	0,8	0,667	0,714	0,625	0,667	0,6	0,636	0,583
Italien	1	1	0,667	0,75	0,8	0,833	0,857	0,75	0,778	0,8	0,818	0,833
Telefon	0	0,5	0,667	0,75	0,8	0,667	0,714	0,75	0,778	0,7	0,727	0,75
Tasche	0	0	0	0	0	0	0	0,125	0,111	0,1	0,182	0,25
Affe	1	1	1	1	1	0,833	0,857	0,875	0,889	0,9	0,909	0,917
Torte	0	0,5	0,333	0,5	0,6	0,667	0,714	0,75	0,667	0,6	0,546	0,583
Platz	0	0	0,333	0,25	0,2	0,333	0,429	0,5	0,556	0,6	0,636	0,667
Sand	0	0	0	0	0	0	0	0	0,111	0,1	0,091	0,083
Kuchen	1	1	1	0,75	0,8	0,667	0,714	0,75	0,667	0,7	0,636	0,583
Boot	1	1	1	1	0,8	0,667	0,714	0,75	0,778	0,7	0,727	0,75
Zebra	0	0	0	0	0	0,167	0,143	0,25	0,333	0,4	0,455	0,5
Bach	0	0,5	0,333	0,5	0,6	0,667	0,571	0,625	0,556	0,5	0,546	0,5
Reifen	0	0	0,333	0,5	0,4	0,333	0,429	0,375	0,444	0,5	0,546	0,583
wedel751	1	1	1	1	1	1	1	1	1	0,9	0,909	0,917
wedel714	1	1	0,667	0,75	0,6	0,667	0,714	0,625	0,667	0,7	0,727	0,75
wedel804	1	0,5	0,667	0,75	0,8	0,667	0,714	0,625	0,667	0,7	0,636	0,583
wedel524	1	1	1	0,75	0,8	0,833	0,714	0,75	0,778	0,8	0,727	0,75
wedel673	1	1	0,667	0,5	0,4	0,5	0,429	0,5	0,556	0,6	0,546	0,5
wedel480	1	1	0,667	0,75	0,8	0,833	0,857	0,875	0,889	0,8	0,727	0,75
wedel723	1	0,5	0,333	0,25	0,4	0,5	0,571	0,625	0,667	0,6	0,546	0,583
wedel836	1	1	1	1	0,8	0,833	0,857	0,75	0,778	0,8	0,818	0,833
wedel563	1	1	1	1	1	1	1	1	1	1	1	1
wedel630	0	0,5	0,667	0,5	0,6	0,5	0,571	0,625	0,556	0,6	0,636	0,667
wedel692	1	1	0,667	0,5	0,4	0,333	0,429	0,5	0,444	0,5	0,546	0,583
wedel579	1	1	0,667	0,5	0,6	0,667	0,714	0,75	0,778	0,7	0,636	0,583
wedel603	1	1	1	0,75	0,6	0,5	0,429	0,5	0,556	0,5	0,546	0,583
wedel862	1	1	0,667	0,5	0,6	0,667	0,714	0,75	0,778	0,7	0,636	0,667
mean	76,06%	72,54%	68,08%	63,38%	61,41%	61,03%	63,38%	64,26%	66,51%	65,07%	65,30%	65,84%
SD	0,427	0,344	0,277	0,271	0,254	0,242	0,226	0,206	0,192	0,181	0,169	0,174
Confidence alpha=0.05	0,099	0,08	0,064	0,063	0,059	0,056	0,053	0,048	0,045	0,042	0,039	0,04
Confidence interval min	66,13%	64,54%	61,64%	57,07%	55,50%	55,40%	58,13%	59,47%	62,04%	60,87%	61,36%	61,81%
Confidence interval max	85,98%	80,53%	74,51%	69,69%	67,32%	66,67%	68,63%	69,05%	70,99%	69,27%	69,24%	69,88%

H.2.5: The table shows for each participant (id) the coherence performance after 92..103 pair comparisons (i.e, the last 12 pair comparisons).

ids	#92	#93	#94	#95	#96	#97	#98	#99	#100	#101	#102	#103
Banane	0,935	0,936	0,936	0,937	0,938	0,938	0,939	0,939	0,94	0,941	0,941	0,942
Eule	0,804	0,807	0,809	0,811	0,813	0,814	0,816	0,818	0,82	0,822	0,824	0,825
Lampe	0,685	0,688	0,692	0,695	0,688	0,691	0,684	0,687	0,68	0,683	0,677	0,67
Feuer	0,859	0,86	0,862	0,863	0,865	0,866	0,867	0,869	0,87	0,871	0,873	0,874
Katze	0,967	0,968	0,968	0,968	0,969	0,969	0,969	0,97	0,97	0,97	0,971	0,971
Blume	0,794	0,796	0,798	0,8	0,802	0,804	0,806	0,808	0,8	0,802	0,804	0,806
Regen	0,859	0,86	0,862	0,853	0,854	0,856	0,857	0,859	0,86	0,861	0,853	0,845
Hund	0,924	0,925	0,926	0,926	0,927	0,928	0,929	0,929	0,93	0,931	0,931	0,932
Wasser	0,87	0,871	0,872	0,874	0,875	0,876	0,878	0,879	0,87	0,871	0,873	0,874
Tiere	0,489	0,495	0,5	0,505	0,5	0,505	0,5	0,495	0,49	0,495	0,5	0,505
Koffer	0,804	0,807	0,809	0,8	0,802	0,794	0,796	0,798	0,8	0,802	0,804	0,806
Igel	0,946	0,946	0,947	0,947	0,948	0,949	0,949	0,95	0,95	0,951	0,951	0,952
Berlin	0,815	0,817	0,809	0,811	0,813	0,814	0,816	0,818	0,82	0,822	0,824	0,825
Bus	0,924	0,925	0,926	0,926	0,927	0,928	0,929	0,929	0,93	0,931	0,931	0,932
Bremen	0,88	0,882	0,883	0,884	0,885	0,887	0,888	0,889	0,89	0,891	0,892	0,893
Frosch	0,837	0,839	0,84	0,842	0,844	0,845	0,847	0,849	0,85	0,852	0,853	0,854
Elbe	0,511	0,505	0,5	0,495	0,49	0,495	0,49	0,495	0,49	0,495	0,49	0,485
Farbe	0,88	0,882	0,883	0,884	0,885	0,887	0,888	0,889	0,88	0,881	0,882	0,884
Hase	0,739	0,731	0,734	0,737	0,729	0,732	0,735	0,737	0,74	0,743	0,745	0,748
Genf	0,62	0,624	0,628	0,632	0,625	0,629	0,622	0,626	0,63	0,634	0,637	0,641
Haus	0,837	0,839	0,84	0,842	0,844	0,845	0,847	0,849	0,85	0,852	0,853	0,854
Eisen	0,761	0,753	0,745	0,747	0,75	0,753	0,755	0,758	0,76	0,753	0,755	0,748
Buch	0,88	0,882	0,883	0,874	0,875	0,876	0,878	0,879	0,88	0,881	0,873	0,864
Kran	0,837	0,839	0,84	0,842	0,844	0,845	0,847	0,849	0,85	0,852	0,853	0,854
Aula	0,913	0,914	0,915	0,916	0,917	0,918	0,918	0,919	0,92	0,921	0,922	0,922
Elch	0,467	0,462	0,457	0,453	0,448	0,443	0,439	0,444	0,45	0,455	0,461	0,456
Golf	0,587	0,591	0,585	0,579	0,583	0,577	0,582	0,586	0,59	0,594	0,598	0,592
Ampel	0,946	0,946	0,947	0,947	0,948	0,949	0,949	0,95	0,95	0,951	0,951	0,952
Biber	0,88	0,882	0,883	0,884	0,885	0,887	0,888	0,889	0,89	0,891	0,892	0,893
Acht	0,554	0,559	0,564	0,568	0,563	0,557	0,551	0,546	0,54	0,545	0,549	0,544
Indien	0,663	0,667	0,67	0,663	0,667	0,67	0,674	0,677	0,68	0,673	0,667	0,66
Post	0,935	0,936	0,936	0,937	0,938	0,938	0,939	0,939	0,94	0,941	0,941	0,942
Tulpe	0,63	0,634	0,638	0,642	0,635	0,639	0,643	0,647	0,64	0,634	0,637	0,641
Mais	0,967	0,968	0,968	0,968	0,969	0,969	0,969	0,97	0,97	0,97	0,971	0,971
Belgien	0,739	0,742	0,745	0,747	0,74	0,742	0,745	0,748	0,75	0,753	0,755	0,757
Italien	0,837	0,839	0,84	0,842	0,844	0,835	0,837	0,838	0,84	0,842	0,843	0,835
Telefon	0,815	0,817	0,819	0,821	0,823	0,814	0,816	0,818	0,82	0,822	0,824	0,825
Tasche	0,544	0,538	0,532	0,537	0,542	0,536	0,541	0,546	0,55	0,545	0,549	0,553
Affe	0,946	0,946	0,947	0,947	0,948	0,949	0,949	0,95	0,95	0,951	0,951	0,952
Torte	0,62	0,624	0,628	0,632	0,635	0,629	0,633	0,626	0,63	0,634	0,637	0,641
Platz	0,75	0,753	0,755	0,758	0,76	0,763	0,755	0,758	0,76	0,762	0,765	0,767
Sand	0,544	0,548	0,553	0,558	0,552	0,557	0,551	0,556	0,55	0,555	0,559	0,563
Kuchen	0,565	0,57	0,575	0,579	0,573	0,577	0,571	0,566	0,56	0,555	0,549	0,544
Boot	0,783	0,785	0,777	0,779	0,781	0,784	0,786	0,778	0,78	0,782	0,784	0,786
Zebra	0,663	0,667	0,67	0,674	0,667	0,67	0,663	0,667	0,67	0,673	0,677	0,68
Bach	0,62	0,613	0,606	0,611	0,615	0,619	0,612	0,616	0,61	0,604	0,608	0,612
Reifen	0,761	0,763	0,766	0,768	0,771	0,773	0,776	0,778	0,78	0,782	0,784	0,786
wedel751	0,946	0,946	0,947	0,947	0,948	0,949	0,949	0,95	0,95	0,951	0,951	0,952
wedel714	0,87	0,871	0,872	0,874	0,875	0,876	0,878	0,879	0,88	0,881	0,882	0,884
wedel804	0,63	0,634	0,628	0,621	0,615	0,619	0,622	0,616	0,62	0,624	0,618	0,612
wedel524	0,783	0,774	0,766	0,768	0,771	0,773	0,765	0,768	0,76	0,762	0,765	0,767
wedel673	0,609	0,613	0,617	0,621	0,625	0,629	0,622	0,626	0,62	0,614	0,608	0,612
wedel480	0,924	0,925	0,926	0,926	0,927	0,928	0,929	0,929	0,93	0,931	0,931	0,932
wedel723	0,707	0,71	0,713	0,716	0,719	0,711	0,704	0,697	0,7	0,703	0,706	0,709
wedel836	0,859	0,86	0,862	0,863	0,865	0,866	0,867	0,869	0,87	0,871	0,873	0,874
wedel563	1	1	1	1	1	1	1	1	1	1	1	1
wedel630	0,728	0,731	0,734	0,737	0,74	0,742	0,745	0,748	0,75	0,753	0,755	0,757
wedel692	0,739	0,742	0,745	0,747	0,75	0,753	0,745	0,748	0,74	0,733	0,735	0,738
wedel579	0,772	0,774	0,777	0,779	0,781	0,784	0,776	0,768	0,77	0,772	0,775	0,777
wedel603	0,804	0,807	0,809	0,811	0,813	0,814	0,806	0,798	0,8	0,802	0,804	0,806
wedel862	0,815	0,817	0,819	0,821	0,823	0,825	0,827	0,828	0,83	0,832	0,833	0,835
Mean	78,37%	78,49%	78,57%	78,67%	78,68%	78,77%	78,72%	78,79%	78,76%	78,86%	78,96%	78,98%
SD	0,134	0,134	0,134	0,134	0,135	0,135	0,136	0,136	0,136	0,136	0,136	0,136
Confidence alpha=0.05	0,031	0,031	0,031	0,031	0,031	0,031	0,032	0,032	0,032	0,032	0,032	0,032
Confidence interval min	75,25%	75,38%	75,46%	75,56%	75,55%	75,64%	75,55%	75,62%	75,59%	75,69%	75,79%	75,81%
Confidence interval max	81,48%	81,61%	81,69%	81,77%	81,82%	81,90%	81,88%	81,96%	81,94%	82,03%	82,12%	82,15%

H2.4 - The table shows the relative coherence judgements for all participants of group 1. For example participant “Seife” judged in the first 18 pair comparisons 0,778 % coherent to her/his own classification. The mean of the quartiles reflect the average coherence for the first, second, third and fourth quarter of judgements.

	Seife	Apfel	Hamburg	Baum	Wolke	Birne	Sonne	Kerze	Licht	Kiel	Banane	Eule	Lampe	Feuer	Katze	Blume	Regen	Hund	Wasser	Tiere	Koffer	Igel
1	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1
2	0,5	1	0,5	0	0	1	1	1	1	0,5	0,5	1	0	0,5	1	1	1	1	0,5	1	0,5	1
3	0,667	1	0,333	0,333	0,333	1	1	1	0,667	0,667	0,667	1	0	0,667	0,667	0,667	1	1	0,667	0,667	0,667	1
4	0,5	1	0,25	0,5	0,5	1	1	1	0,5	0,5	0,75	0,75	0	0,75	0,5	0,5	1	1	0,5	0,75	0,75	1
5	0,4	1	0,2	0,6	0,4	0,8	1	1	0,4	0,6	0,8	0,6	0,2	0,6	0,4	0,4	1	1	0,6	0,8	0,6	0,8
6	0,5	0,833	0,167	0,667	0,333	0,833	1	1	0,5	0,667	0,667	0,5	0,167	0,5	0,5	0,333	1	1	0,667	0,667	0,5	0,833
7	0,571	0,857	0,286	0,571	0,429	0,857	1	0,857	0,571	0,571	0,714	0,571	0,286	0,571	0,571	0,429	1	1	0,714	0,714	0,571	0,857
8	0,625	0,875	0,25	0,5	0,375	0,875	0,875	0,75	0,625	0,5	0,75	0,5	0,375	0,5	0,625	0,5	1	0,875	0,75	0,75	0,625	0,875
9	0,667	0,889	0,333	0,556	0,444	0,889	0,889	0,778	0,667	0,556	0,778	0,556	0,444	0,556	0,667	0,556	1	0,889	0,778	0,778	0,667	0,889
10	0,7	0,9	0,3	0,6	0,4	0,9	0,8	0,7	0,7	0,5	0,7	0,5	0,5	0,5	0,7	0,6	1	0,9	0,8	0,7	0,6	0,9
11	0,727	0,909	0,364	0,636	0,455	0,818	0,818	0,636	0,636	0,455	0,727	0,546	0,546	0,546	0,727	0,636	1	0,909	0,727	0,727	0,546	0,818
12	0,75	0,917	0,333	0,583	0,5	0,833	0,833	0,583	0,667	0,5	0,75	0,583	0,5	0,583	0,75	0,667	1	0,917	0,75	0,667	0,583	0,833
13	0,769	0,923	0,385	0,615	0,539	0,846	0,846	0,615	0,692	0,539	0,769	0,615	0,539	0,539	0,769	0,692	1	0,846	0,769	0,692	0,615	0,846
14	0,786	0,857	0,429	0,571	0,5	0,857	0,786	0,643	0,714	0,571	0,786	0,643	0,571	0,571	0,786	0,714	0,929	0,857	0,786	0,643	0,643	0,857
15	0,8	0,867	0,467	0,533	0,467	0,867	0,733	0,667	0,733	0,6	0,8	0,6	0,533	0,6	0,8	0,733	0,933	0,867	0,8	0,6	0,6	0,867
16	0,813	0,875	0,5	0,563	0,5	0,875	0,75	0,688	0,75	0,625	0,813	0,625	0,563	0,563	0,813	0,75	0,938	0,813	0,813	0,625	0,625	0,875
17	0,765	0,882	0,471	0,588	0,471	0,882	0,706	0,647	0,706	0,647	0,765	0,647	0,588	0,529	0,824	0,706	0,882	0,765	0,765	0,647	0,647	0,824
18	0,778	0,889	0,5	0,611	0,5	0,889	0,722	0,667	0,722	0,667	0,778	0,667	0,611	0,556	0,833	0,722	0,889	0,778	0,778	0,667	0,667	0,833
19	0,79	0,895	0,526	0,579	0,474	0,895	0,684	0,684	0,737	0,684	0,79	0,632	0,632	0,579	0,842	0,737	0,895	0,79	0,79	0,632	0,684	0,842
20	0,8	0,9	0,5	0,6	0,45	0,9	0,7	0,7	0,7	0,7	0,8	0,6	0,6	0,55	0,85	0,7	0,9	0,8	0,8	0,65	0,65	0,8
21	0,81	0,905	0,524	0,619	0,476	0,905	0,714	0,714	0,714	0,714	0,81	0,571	0,619	0,571	0,857	0,714	0,857	0,81	0,81	0,619	0,667	0,81
22	0,818	0,864	0,5	0,591	0,455	0,909	0,682	0,682	0,682	0,682	0,818	0,546	0,636	0,546	0,864	0,682	0,818	0,773	0,773	0,636	0,636	0,773
23	0,826	0,826	0,478	0,565	0,478	0,913	0,696	0,696	0,652	0,696	0,826	0,565	0,652	0,565	0,87	0,652	0,826	0,783	0,739	0,652	0,609	0,783
24	0,833	0,792	0,5	0,542	0,5	0,917	0,667	0,708	0,667	0,667	0,833	0,542	0,667	0,583	0,875	0,667	0,833	0,792	0,75	0,625	0,625	0,792
25	0,84	0,8	0,48	0,52	0,52	0,92	0,68	0,72	0,64	0,68	0,84	0,56	0,68	0,6	0,88	0,64	0,84	0,8	0,72	0,6	0,64	0,8
26	0,846	0,769	0,5	0,539	0,539	0,923	0,692	0,731	0,654	0,692	0,846	0,539	0,654	0,615	0,885	0,615	0,846	0,808	0,731	0,615	0,615	0,808
Median 1.Quart	0,688	0,893	0,426	0,522	0,424	0,896	0,818	0,764	0,681	0,622	0,772	0,633	0,445	0,586	0,764	0,654	0,938	0,876	0,703	0,697	0,636	0,858
27	0,852	0,778	0,519	0,556	0,556	0,926	0,704	0,741	0,667	0,667	0,852	0,556	0,667	0,63	0,889	0,63	0,852	0,815	0,741	0,593	0,63	0,815
28	0,857	0,786	0,536	0,536	0,536	0,929	0,714	0,75	0,679	0,679	0,857	0,571	0,679	0,643	0,893	0,643	0,857	0,786	0,75	0,607	0,607	0,821
29	0,862	0,793	0,552	0,552	0,552	0,931	0,724	0,759	0,69	0,655	0,862	0,586	0,69	0,655	0,897	0,655	0,862	0,793	0,759	0,621	0,621	0,828
30	0,867	0,8	0,567	0,567	0,567	0,933	0,733	0,767	0,7	0,667	0,867	0,6	0,7	0,667	0,9	0,667	0,867	0,8	0,767	0,6	0,633	0,833
31	0,871	0,807	0,581	0,581	0,581	0,936	0,742	0,774	0,71	0,677	0,871	0,613	0,71	0,677	0,903	0,677	0,871	0,807	0,774	0,581	0,645	0,839
32	0,875	0,813	0,563	0,563	0,563	0,938	0,75	0,781	0,719	0,656	0,875	0,625	0,688	0,688	0,906	0,688	0,844	0,813	0,781	0,563	0,625	0,844
33	0,879	0,818	0,546	0,576	0,576	0,909	0,758	0,788	0,697	0,636	0,879	0,606	0,667	0,697	0,909	0,697	0,849	0,818	0,758	0,546	0,606	0,849
34	0,882	0,824	0,559	0,588	0,588	0,912	0,765	0,794	0,706	0,647	0,882	0,618	0,677	0,706	0,912	0,706	0,853	0,824	0,765	0,559	0,618	0,853
35	0,886	0,829	0,571	0,6	0,6	0,914	0,771	0,8	0,714	0,657	0,886	0,629	0,686	0,714	0,914	0,714	0,829	0,829	0,771	0,543	0,629	0,857
36	0,889	0,833	0,583	0,611	0,611	0,917	0,778	0,806	0,722	0,639	0,889	0,639	0,694	0,722	0,917	0,722	0,833	0,833	0,778	0,556	0,611	0,861
37	0,892	0,811	0,595	0,622	0,622	0,919	0,784	0,811	0,73	0,649	0,892	0,622	0,703	0,73	0,919	0,703	0,838	0,838	0,784	0,541	0,622	0,865
38	0,895	0,816	0,605	0,632	0,632	0,921	0,79	0,816	0,737	0,658	0,895	0,632	0,711	0,737	0,921	0,711	0,816	0,842	0,79	0,553	0,632	0,868
39	0,897	0,821	0,615	0,641	0,641	0,923	0,795	0,821	0,744	0,667	0,897	0,641	0,718	0,744	0,923	0,718	0,821	0,846	0,795	0,564	0,641	0,872
40	0,9	0,825	0,6	0,65	0,65	0,925	0,8	0,825	0,75	0,675	0,9	0,625	0,725	0,75	0,925	0,725	0,825	0,85	0,8	0,55	0,65	0,875
41	0,902	0,829	0,61	0,659	0,659	0,927	0,805	0,829	0,756	0,683	0,902	0,634	0,732	0,756	0,927	0,732	0,829	0,854	0,805	0,537	0,659	0,878
42	0,905	0,81	0,619	0,667	0,667	0,929	0,81	0,833	0,762	0,691	0,905	0,643	0,738	0,762	0,929	0,714	0,833	0,857	0,81	0,548	0,667	0,881
43	0,907	0,814	0,605	0,674	0,674	0,93	0,814	0,837	0,767	0,674	0,907	0,651	0,744	0,767	0,93	0,721	0,837	0,861	0,814	0,535	0,674	0,884
44	0,909	0,796	0,591	0,659	0,682	0,932	0,818	0,841	0,773	0,659	0,909	0,636	0,75	0,773	0,932	0,727	0,841	0,864	0,818	0,523	0,682	0,886
45	0,911	0,8	0,578	0,644	0,689	0,933	0,822	0,844	0,778	0,644	0,911	0,644	0,756	0,778	0,933	0,711	0,844	0,867	0,8	0,511	0,667	0,889
46	0,913	0,804	0,587	0,652	0,696	0,935	0,826	0,848	0,783	0,652	0,913	0,652	0,761	0,783	0,935	0,717	0,848	0,87	0,804	0,522	0,674	0,891
47	0,915	0,809	0,575	0,638	0,702	0,936	0,83	0,851	0,787	0,638	0,915	0,66	0,766	0,787	0,936	0,723	0,851	0,872	0,809	0,511	0,66	0,894
48	0,917	0,813	0,583	0,646	0,708	0,938	0,833	0,854	0,792	0,646	0,917	0,667	0,771	0,792	0,938	0,729	0,854	0,875	0,813	0,521	0,667	0,896
49	0,918	0,796	0,571	0,633	0,714	0,939	0,837	0,857	0,796	0,633	0,918	0,674	0,755	0,796	0,939	0,735	0,857	0,878	0,816	0,51	0,674	0,898
50	0,92	0,8	0,58	0,64	0,72	0,94	0,84	0,86	0,8	0,64	0,92	0,68	0,74	0,8	0,94	0,74	0,84	0,88	0,82	0,52	0,68	0,9
51	0,922	0,804	0,588	0,647	0,726	0,941	0,843	0,863	0,804	0,628	0,922	0,686	0,726	0,804	0,941	0,745	0,843	0,882	0,824	0,51	0,686	0,902
52	0,923	0,808	0,596	0,654	0,731	0,942	0,846	0,865	0,808	0,635	0,923	0,692	0,731	0,808	0,942	0,75	0,846	0,885	0,827	0,519	0,692	0,904
Mean2. Quart	0,895	0,809	0,58	0,619	0,64	0,929	0,79	0,816	0,745	0,656	0,895	0,634	0,718	0,737	0,921	0,708	0,844	0,844	0,791	0,548	0,648	0,869
53	0,925	0,811	0,604	0,66	0,736	0,943	0,849	0,868	0,811	0,642	0,925	0,698	0,736	0,811	0,943	0,755	0,849	0,887	0,83	0,528	0,698	0,906
54	0,926	0,815	0,611	0,667	0,722	0,944	0,833	0,87	0,815	0,648	0,926	0,704	0,722	0,815	0,944	0,759	0,852	0,889	0,833	0,519	0,704	0,907
55	0,927	0,818	0,618	0,673	0,727	0,946	0,836	0,873	0,818	0,655	0,927	0,709	0,727	0,818	0,946	0,764	0,855	0,891	0,836	0,527	0,709	0,909
56	0,929	0,821	0,625	0,679	0,732	0,946	0,839	0,875	0,821	0,661	0,929	0,714	0,732	0,821	0,946	0,768	0,857	0,893	0,839	0,536	0,714	0,911
57	0,93	0,825	0,632	0,684	0,737	0,947	0,842	0,877	0,825	0,667	0,93	0,719	0,737	0,825	0,947	0,772	0,86	0,895	0,842	0,544	0,719	0,912
58	0,931	0,828	0,638	0,69	0,741	0,948	0,845	0,879	0,828	0,672	0,931	0,724	0,741	0,828	0,948	0,776	0,862	0,897	0,845	0,552	0,724	0,914
59	0,932	0,831	0,627	0,678	0,746	0,949	0,848	0,881	0,831	0,678	0,932	0,712	0,746	0,831	0,949	0,78	0,864	0,898	0,848	0,542	0,729	0,915
60	0,933	0,833	0,633	0,683	0,75	0,933	0,85	0,883	0,833	0,683	0,933	0,7	0,75	0,833	0,95	0,767	0,85	0,9	0,85	0,533	0,733	0,917
61	0,934	0,836	0,623	0,672	0,738	0,934	0,836	0,869	0,836	0,689	0,918	0,705	0,738	0,82	0,951	0,771	0,853	0,885	0,853	0,525	0,721	0,918
62	0,936	0,839	0,629	0,677	0,742	0,936	0,823	0,871	0,839	0,694	0,919	0,71	0,726	0,823	0,952	0,774	0,839	0,887	0,855	0,532	0,726	0,919
63	0,937	0,841	0,635	0,683	0,746	0,937	0,825	0,873	0,841	0,698	0,921	0,714	0,73	0,825	0,952	0,778	0,841	0,889	0,857	0,54	0,73	0,921
64	0,938	0,844	0,625	0,688	0,75	0,922	0,828	0,875	0,844	0,703	0,922	0,719	0,734	0,828	0,953	0,781	0,844	0,891	0,859	0,531	0,734	0,922
65	0,939	0,846	0,631	0,692	0,754	0,923	0,831	0,877	0,846	0,708	0,923	0,723	0,739	0,831	0,954	0,785	0,846	0,892	0,862	0,539	0,739	0,923
66	0,939	0,849	0,636	0,697	0,758	0,924	0,833	0,879	0,849	0,712	0,924	0,727	0,727	0,833	0,955	0,788	0,849	0,894	0,864	0,53	0,742	0,924
67	0,94	0,851	0,642	0,702	0,761	0,925	0,836	0,881	0,851	0,702	0,925	0,731	0,731	0,836	0,955	0,791	0,851	0,896	0,866	0,537	0,746	0,925
68	0,941	0,853	0,632	0,691	0,765	0,927	0,838	0,882	0,853	0,691	0,927	0,735	0,735	0,838	0,956	0,794	0,838	0,897	0,868	0,529	0,75	0,927
69	0,942	0,855	0,623	0,696	0,768	0,928	0,841	0,884	0,855	0,681	0,928	0,739	0,739	0,841	0,957	0,797	0,841	0,899	0,87	0,536	0,754	0,928
70	0,943	0,857	0,629	0,686	0,771	0,929	0,843	0,886	0,857	0,686	0,929	0,743	0,743	0,843	0,957	0,8	0,843	0,9	0,871	0,543	0,757	0,929
71	0,944	0,859	0,634	0,69	0,775	0,93	0,845	0,887	0,859	0,69	0,93	0,747	0,747	0,845	0,958	0,803	0,845	0,901	0,873	0,549	0,761	0,93
72	0,944	0,847	0,625	0,681	0,778	0,917	0,847	0,889	0,861	0,694	0,931	0,75	0,75	0,847	0,958	0,792	0,847	0,903	0,861	0,542	0,764	0,931
73	0,945	0,849	0,63	0,671	0,781	0,918	0,849	0,89	0,863	0,685	0,932	0,753	0,74	0,849	0,959	0,795	0,836	0,904	0,863	0,548	0,767	0,932
74	0,946	0,851	0,622	0,662	0,784	0,919	0,851	0,892	0,865	0,689	0,932	0,757	0,73	0,851	0,96	0,784	0,838	0,905	0,865	0,541	0,77	0,932
75	0,947	0,853	0,627	0,653	0,787	0,92	0,853	0,893	0,867	0,68	0,933	0,76	0,72	0,853	0,96	0,773	0,84	0,907	0,853	0,533	0,773	0,933
76	0,947	0,855	0,632	0,658	0,79	0,921	0,855	0,895	0,855	0,671	0,934	0,763	0,711	0,855	0,961	0,776	0,842	0,908	0,855	0,526	0,776	0,934
77	0,948	0,857	0,636	0,662	0,792	0,922	0,857	0,896	0,857	0,662	0,935	0,766	0,714	0,857	0,961	0,779	0,844	0,909	0,857	0,533	0,779	0,935
78	0,949	0,859	0,641	0,667	0,795	0,923	0,859	0,897	0,859	0,667	0,936	0,769	0,705	0,859	0,962	0,782	0,846	0,91	0,859	0,526	0,782	0,936
Mean3. Quart	0,938	0,842	0,628	0,678	0,759	0,931	0,842	0,882	0,844	0,681	0,928	0,73	0,733	0,835	0,954	0,78	0,847	0,897	0,855	0,535	0,742	0,923
79	0,949	0,861	0,646	0,671	0,798	0,911	0,848	0,899	0,861	0,671	0,937	0,772	0,709	0,848	0,962	0,772	0,848	0,911	0,861	0,519	0,785	0,937
80	0,95	0,863	0,65	0,675	0,8	0,913	0,85	0,9	0,85	0,675	0,938	0,775	0,7	0,85	0,963	0,763	0,85	0,913	0,85	0,513	0,788	0,938
81	0,951	0,864	0,654	0,679	0,803	0,914	0,852	0,901	0,852	0,667	0,938	0,778	0,704	0,852	0,963	0,765	0,84	0,914	0,852	0,506	0,79	0,938
82	0,951	0,866	0,659	0,683	0,805	0,915	0,854	0,902	0,854	0,671	0,939	0,781	0,695	0,854	0,963	0,768	0,842	0,915	0,854	0,512	0,793	0,939
83	0,952	0,868	0,663	0,675	0,807	0,916	0,855	0,904	0,855	0,675	0,94	0,783	0,699	0,855	0,964	0,771	0,843	0,916	0,855	0,506	0,795	0,94
84	0,952	0,869	0,667	0,679	0,81	0,917	0,857	0,905	0,857	0,679	0,941	0,786	0,702	0,845	0,964	0,774	0,845	0,917	0,857	0,5	0,798	0,941
85	0,953	0,871	0,659	0,682	0,812	0,918	0,847	0,906	0,859	0,682	0,941	0,788	0,694	0,847	0,965	0,777	0,847	0,918	0,859	0,506	0,8	0,941
86	0,954	0,872	0,651	0,686	0,814	0,919	0,849	0,907	0,861	0,674	0,942	0,791	0,686	0,849	0,965	0,779	0,849	0,919	0,861	0,5	0,802	0,942
87	0,954	0,874	0,655	0,69	0,816	0,92	0,851	0,908	0,862	0,678	0,943	0,793	0,678	0,851	0,966	0,782	0,851	0,92	0,862	0,494	0,805	0,943
88	0,955	0,875	0,659	0,693	0,818	0,921	0,852	0,909	0,864	0,682	0,943	0,796	0,682	0,852	0,966	0,784	0,852	0,921	0,864	0,5	0,807	0,943
89	0,955	0,876	0,663	0,697	0,82	0,921	0,854	0,91	0,865	0,685	0,944	0,798	0,685	0,854	0,966	0,787	0,854	0,921	0,865	0,494	0,809	0,944
90	0,956	0,878	0,667	0,7	0,822	0,922	0,856	0,911	0,867	0,689	0,944	0,8	0,689	0,856	0,967	0,789	0,856	0,922	0,867	0,5	0,811	0,944
91	0,956	0,879	0,659	0,703	0,824	0,923	0,857	0,912	0,868	0,681	0,945	0,802	0,681	0,857	0,967	0,791	0,857	0,923	0,868	0,495	0,802	0,945
92	0,957	0,88	0,663	0,707	0,826	0,924	0,848	0,913	0,87	0,685	0,935	0,804	0,685	0,859	0,967	0,794	0,859	0,924	0,87	0,489	0,804	0,946
93	0,957	0,882	0,667	0,71	0,828	0,925	0,85	0,914	0,871	0,688	0,936	0,807	0,688	0,86	0,968	0,796	0,86	0,925	0,871	0,495	0,807	0,946
94	0,957	0,883	0,67	0,713	0,83	0,926	0,851	0,915	0,872	0,692	0,936	0,809	0,692	0,862	0,968	0,798	0,862	0,926	0,872	0,5	0,809	0,947
95	0,958	0,884	0,674	0,716	0,832	0,926	0,842	0,916	0,874	0,695	0,937	0,811	0,695	0,863	0,968	0,8	0,853	0,926	0,874	0,505	0,8	0,947
96	0,958	0,885	0,667	0,719	0,833	0,927	0,844	0,917	0,875	0,698	0,938	0,813	0,688	0,865	0,969	0,802	0,854	0,927	0,875	0,5	0,802	0,948
97	0,959	0,887	0,67	0,722	0,835	0,928	0,845	0,918	0,876	0,701	0,938	0,814	0,691	0,866	0,969	0,804	0,856	0,928	0,876	0,505	0,794	0,949
98	0,959	0,888	0,663	0,714	0,837	0,929	0,847	0,918	0,878	0,704	0,939	0,816	0,684	0,867	0,969	0,806	0,857	0,929	0,878	0,5	0,796	0,949
99	0,96	0,889	0,657	0,717	0,838	0,929	0,849	0,919	0,879	0,707	0,939	0,818	0,687	0,869	0,97	0,808	0,859	0,929	0,879	0,495	0,798	0,95
100	0,96	0,89	0,66	0,71	0,84	0,93	0,84	0,92	0,87	0,71	0,94	0,82	0,68	0,87	0,97	0,8	0,86	0,93	0,87	0,49	0,8	0,95
101	0,96	0,891	0,663	0,713	0,842	0,931	0,842	0,921	0,871	0,713	0,941	0,822	0,683	0,871	0,97	0,802	0,861	0,931	0,871	0,495	0,802	0,951
102	0,961	0,892	0,667	0,716	0,843	0,931	0,843	0,922	0,873	0,716	0,941	0,824	0,677	0,873	0,971	0,804	0,853	0,931	0,873	0,5	0,804	0,951
103	0,961	0,893	0,67	0,718	0,845	0,932	0,845	0,922	0,874	0,709	0,942	0,825	0,67	0,874	0,971	0,806	0,845	0,932	0,874	0,505	0,806	0,952
Mean4. Quart	0,956	0,878	0,662	0,699	0,823	0,923	0,849	0,912	0,866	0,689	0,94	0,801	0,689	0,859	0,967	0,789	0,852	0,923	0,866	0,501	0,8	0,945

The table shows the relative coherence judgements for all participants of group 2.

	Berlin	Bus	Bremen	Frosch	Elbe	Farbe	Hase	Genf	Haus	Eisen	Buch	Kran	Aula	Elch	Golf	Ampel	Biber
1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1
2	0,5	1	0,5	0,5	0,5	1	1	0,5	1	0,5	1	1	0,5	1	0,5	0,5	1
3	0,333	1	0,667	0,333	0,667	0,667	0,667	0,667	0,667	0,667	1	1	0,667	0,667	0,667	0,667	1
4	0,25	1	0,75	0,25	0,5	0,5	0,75	0,5	0,5	0,75	1	1	0,5	0,5	0,5	0,75	0,75
5	0,4	1	0,6	0,4	0,4	0,4	0,8	0,6	0,4	0,6	1	1	0,6	0,4	0,4	0,8	0,6
6	0,5	1	0,5	0,5	0,333	0,5	0,833	0,5	0,333	0,667	1	1	0,667	0,333	0,333	0,833	0,667
7	0,571	1	0,571	0,571	0,429	0,571	0,714	0,571	0,429	0,714	1	1	0,571	0,286	0,286	0,857	0,571
8	0,625	0,875	0,625	0,625	0,5	0,625	0,625	0,625	0,5	0,75	1	1	0,625	0,375	0,25	0,875	0,5
9	0,667	0,889	0,667	0,667	0,444	0,667	0,667	0,667	0,556	0,667	1	1	0,667	0,333	0,333	0,889	0,556
10	0,6	0,8	0,6	0,7	0,5	0,7	0,6	0,6	0,5	0,7	0,9	0,9	0,7	0,3	0,4	0,8	0,6
11	0,546	0,818	0,636	0,636	0,455	0,636	0,636	0,546	0,546	0,727	0,909	0,909	0,727	0,364	0,455	0,818	0,636
12	0,583	0,833	0,667	0,667	0,417	0,667	0,583	0,5	0,583	0,75	0,917	0,917	0,75	0,333	0,417	0,833	0,667
13	0,615	0,846	0,692	0,692	0,462	0,692	0,615	0,539	0,615	0,769	0,846	0,923	0,769	0,385	0,462	0,846	0,615
14	0,643	0,857	0,714	0,714	0,429	0,714	0,571	0,571	0,571	0,786	0,857	0,929	0,786	0,357	0,5	0,857	0,643
15	0,6	0,867	0,733	0,733	0,467	0,733	0,533	0,533	0,6	0,8	0,867	0,867	0,8	0,333	0,467	0,867	0,667
16	0,625	0,875	0,75	0,75	0,438	0,75	0,563	0,563	0,625	0,813	0,875	0,875	0,813	0,375	0,438	0,875	0,688
17	0,588	0,824	0,706	0,706	0,412	0,706	0,529	0,529	0,588	0,765	0,824	0,824	0,765	0,412	0,412	0,824	0,647
18	0,611	0,833	0,722	0,722	0,444	0,722	0,556	0,556	0,611	0,778	0,833	0,833	0,778	0,444	0,389	0,833	0,667
19	0,632	0,842	0,737	0,737	0,474	0,737	0,579	0,526	0,632	0,737	0,842	0,842	0,79	0,421	0,368	0,842	0,684
20	0,6	0,85	0,75	0,7	0,45	0,7	0,6	0,55	0,65	0,75	0,85	0,85	0,75	0,4	0,35	0,85	0,65
21	0,619	0,857	0,762	0,714	0,476	0,714	0,619	0,571	0,667	0,762	0,857	0,857	0,762	0,381	0,381	0,857	0,667
22	0,591	0,818	0,727	0,682	0,5	0,682	0,591	0,591	0,636	0,773	0,818	0,818	0,727	0,364	0,364	0,818	0,636
23	0,565	0,826	0,696	0,652	0,478	0,652	0,565	0,609	0,652	0,739	0,826	0,826	0,739	0,391	0,348	0,826	0,609
24	0,583	0,833	0,708	0,667	0,458	0,667	0,542	0,625	0,625	0,708	0,833	0,833	0,75	0,375	0,333	0,833	0,625
25	0,6	0,84	0,68	0,68	0,48	0,64	0,56	0,64	0,64	0,72	0,84	0,8	0,76	0,36	0,36	0,84	0,64
26	0,577	0,846	0,692	0,654	0,5	0,654	0,577	0,615	0,654	0,731	0,846	0,808	0,769	0,385	0,385	0,846	0,615
Means 1. Quartil	0,578	0,886	0,687	0,602	0,485	0,681	0,649	0,588	0,607	0,735	0,905	0,908	0,682	0,434	0,427	0,786	0,677
27	0,593	0,852	0,704	0,667	0,482	0,667	0,556	0,63	0,667	0,741	0,852	0,815	0,778	0,407	0,407	0,852	0,63
28	0,571	0,857	0,714	0,643	0,5	0,679	0,571	0,607	0,679	0,714	0,857	0,821	0,786	0,393	0,393	0,857	0,643
29	0,586	0,862	0,724	0,655	0,483	0,69	0,586	0,621	0,69	0,724	0,862	0,828	0,793	0,379	0,379	0,862	0,655
30	0,6	0,867	0,733	0,667	0,467	0,7	0,6	0,633	0,7	0,733	0,867	0,833	0,8	0,4	0,367	0,867	0,667
31	0,613	0,871	0,742	0,677	0,484	0,71	0,581	0,613	0,71	0,71	0,871	0,839	0,807	0,387	0,355	0,871	0,677
32	0,625	0,844	0,75	0,656	0,469	0,719	0,563	0,594	0,719	0,719	0,875	0,844	0,813	0,406	0,375	0,875	0,688
33	0,606	0,818	0,727	0,636	0,455	0,727	0,546	0,576	0,727	0,727	0,879	0,849	0,818	0,394	0,394	0,879	0,667
34	0,618	0,824	0,735	0,647	0,441	0,735	0,559	0,559	0,735	0,735	0,882	0,853	0,824	0,412	0,382	0,882	0,677
35	0,629	0,829	0,714	0,657	0,429	0,743	0,571	0,571	0,743	0,743	0,886	0,829	0,829	0,4	0,4	0,886	0,686
36	0,639	0,833	0,722	0,667	0,444	0,75	0,583	0,583	0,722	0,75	0,889	0,833	0,833	0,417	0,417	0,889	0,694
37	0,622	0,838	0,73	0,676	0,432	0,757	0,595	0,595	0,703	0,757	0,892	0,838	0,811	0,432	0,405	0,892	0,703
38	0,632	0,842	0,737	0,684	0,447	0,763	0,605	0,605	0,711	0,763	0,895	0,842	0,816	0,421	0,395	0,895	0,711
39	0,641	0,846	0,744	0,692	0,462	0,769	0,615	0,615	0,718	0,769	0,897	0,846	0,821	0,41	0,41	0,897	0,718
40	0,65	0,85	0,75	0,7	0,45	0,775	0,625	0,625	0,725	0,775	0,9	0,85	0,825	0,4	0,425	0,9	0,725
41	0,659	0,854	0,756	0,707	0,463	0,781	0,61	0,634	0,732	0,756	0,902	0,829	0,829	0,415	0,439	0,902	0,732
42	0,667	0,857	0,762	0,714	0,452	0,786	0,619	0,643	0,738	0,762	0,905	0,833	0,833	0,405	0,429	0,905	0,738
43	0,674	0,861	0,767	0,721	0,442	0,791	0,628	0,651	0,744	0,744	0,907	0,837	0,837	0,419	0,442	0,907	0,744
44	0,659	0,864	0,773	0,705	0,432	0,796	0,636	0,659	0,75	0,727	0,909	0,841	0,841	0,432	0,455	0,909	0,75
45	0,667	0,867	0,778	0,711	0,422	0,8	0,644	0,667	0,756	0,711	0,911	0,844	0,844	0,422	0,467	0,911	0,756
46	0,674	0,87	0,783	0,717	0,413	0,804	0,652	0,674	0,761	0,717	0,913	0,848	0,848	0,413	0,457	0,913	0,761
47	0,681	0,872	0,787	0,702	0,426	0,809	0,638	0,66	0,766	0,702	0,915	0,851	0,851	0,426	0,468	0,915	0,766
48	0,688	0,875	0,792	0,708	0,438	0,813	0,646	0,667	0,771	0,708	0,917	0,854	0,854	0,438	0,458	0,917	0,771
49	0,694	0,878	0,796	0,714	0,429	0,816	0,653	0,653	0,776	0,714	0,918	0,857	0,857	0,429	0,469	0,918	0,776
50	0,7	0,88	0,8	0,72	0,44	0,82	0,66	0,66	0,78	0,72	0,92	0,86	0,86	0,42	0,48	0,92	0,78
51	0,706	0,882	0,804	0,726	0,431	0,824	0,647	0,667	0,784	0,706	0,922	0,863	0,863	0,431	0,49	0,922	0,784
52	0,712	0,885	0,808	0,731	0,442	0,827	0,654	0,654	0,789	0,712	0,923	0,865	0,865	0,423	0,5	0,923	0,789
Means 2. Quartil	0,646	0,857	0,755	0,689	0,449	0,763	0,609	0,627	0,734	0,732	0,895	0,842	0,828	0,413	0,425	0,895	0,719
53	0,717	0,887	0,811	0,736	0,453	0,83	0,66	0,66	0,793	0,717	0,925	0,868	0,868	0,415	0,491	0,925	0,793
54	0,722	0,889	0,815	0,741	0,444	0,833	0,667	0,667	0,796	0,722	0,926	0,87	0,87	0,426	0,482	0,926	0,796
55	0,727	0,891	0,818	0,746	0,455	0,836	0,673	0,673	0,8	0,727	0,927	0,873	0,873	0,436	0,491	0,927	0,8
56	0,732	0,893	0,821	0,75	0,446	0,839	0,679	0,679	0,804	0,714	0,911	0,857	0,875	0,446	0,5	0,929	0,804
57	0,719	0,895	0,825	0,754	0,456	0,842	0,684	0,667	0,807	0,719	0,912	0,86	0,877	0,456	0,509	0,93	0,807
58	0,724	0,897	0,828	0,759	0,466	0,845	0,69	0,672	0,81	0,724	0,914	0,862	0,879	0,466	0,517	0,931	0,81
59	0,729	0,898	0,831	0,746	0,475	0,848	0,695	0,661	0,814	0,712	0,915	0,864	0,881	0,458	0,525	0,932	0,814
60	0,733	0,9	0,833	0,75	0,467	0,85	0,7	0,65	0,8	0,717	0,917	0,867	0,883	0,45	0,533	0,933	0,817
61	0,738	0,885	0,82	0,754	0,475	0,853	0,705	0,639	0,787	0,721	0,902	0,853	0,869	0,459	0,525	0,918	0,82
62	0,742	0,887	0,823	0,758	0,484	0,855	0,71	0,645	0,79	0,726	0,903	0,855	0,871	0,468	0,532	0,919	0,823
63	0,746	0,889	0,825	0,762	0,492	0,857	0,714	0,651	0,794	0,73	0,889	0,857	0,873	0,46	0,54	0,921	0,825
64	0,75	0,891	0,828	0,766	0,5	0,859	0,719	0,641	0,797	0,734	0,891	0,859	0,875	0,453	0,547	0,922	0,828
65	0,754	0,892	0,831	0,769	0,508	0,862	0,708	0,631	0,8	0,723	0,892	0,846	0,877	0,462	0,554	0,923	0,831
66	0,758	0,894	0,833	0,773	0,515	0,864	0,712	0,621	0,803	0,727	0,894	0,849	0,879	0,455	0,561	0,924	0,833
67	0,761	0,896	0,836	0,776	0,522	0,866	0,716	0,627	0,806	0,731	0,896	0,851	0,881	0,448	0,567	0,925	0,836
68	0,765	0,897	0,838	0,779	0,529	0,868	0,721	0,632	0,809	0,735	0,897	0,838	0,882	0,441	0,559	0,927	0,838
69	0,768	0,899	0,841	0,783	0,522	0,87	0,725	0,638	0,812	0,739	0,899	0,841	0,884	0,449	0,551	0,928	0,841
70	0,771	0,9	0,843	0,786	0,514	0,871	0,729	0,643	0,814	0,743	0,9	0,829	0,886	0,457	0,543	0,929	0,843
71	0,775	0,901	0,845	0,789	0,521	0,873	0,732	0,634	0,817	0,747	0,901	0,831	0,887	0,465	0,535	0,93	0,845
72	0,778	0,903	0,847	0,792	0,528	0,875	0,736	0,625	0,819	0,75	0,903	0,833	0,889	0,458	0,528	0,931	0,847
73	0,781	0,904	0,849	0,795	0,521	0,877	0,74	0,616	0,822	0,753	0,904	0,822	0,89	0,466	0,521	0,932	0,849
74	0,784	0,905	0,851	0,797	0,527	0,878	0,743	0,608	0,824	0,757	0,905	0,824	0,892	0,46	0,527	0,932	0,851
75	0,787	0,907	0,853	0,8	0,533	0,88	0,747	0,6	0,827	0,76	0,907	0,827	0,893	0,467	0,533	0,933	0,853
76	0,79	0,908	0,855	0,803	0,54	0,882	0,75	0,605	0,829	0,763	0,908	0,829	0,895	0,461	0,54	0,934	0,855
77	0,792	0,909	0,857	0,805	0,546	0,883	0,753	0,597	0,831	0,766	0,896	0,831	0,896	0,468	0,546	0,935	0,857
78	0,795	0,91	0,859	0,808	0,551	0,885	0,756	0,603	0,833	0,756	0,897	0,833	0,897	0,462	0,551	0,936	0,859
Means 3. Quartil	0,755	0,897	0,835	0,772	0,5	0,861	0,714	0,638	0,809	0,735	0,905	0,847	0,882	0,454	0,531	0,928	0,83
79	0,798	0,911	0,861	0,81	0,557	0,886	0,76	0,595	0,823	0,76	0,899	0,835	0,899	0,456	0,557	0,937	0,861
80	0,8	0,913	0,863	0,813	0,563	0,875	0,75	0,588	0,825	0,763	0,9	0,838	0,9	0,45	0,563	0,938	0,863
81	0,79	0,914	0,864	0,815	0,556	0,877	0,753	0,593	0,827	0,765	0,889	0,84	0,901	0,457	0,568	0,938	0,864
82	0,793	0,915	0,866	0,817	0,549	0,878	0,744	0,598	0,829	0,768	0,89	0,842	0,902	0,463	0,573	0,939	0,866
83	0,795	0,916	0,868	0,819	0,542	0,88	0,747	0,602	0,831	0,759	0,88	0,831	0,904	0,47	0,578	0,94	0,868
84	0,798	0,917	0,869	0,821	0,536	0,881	0,75	0,607	0,821	0,762	0,881	0,833	0,905	0,476	0,583	0,941	0,869
85	0,8	0,918	0,871	0,824	0,541	0,882	0,753	0,612	0,824	0,765	0,882	0,824	0,906	0,471	0,577	0,941	0,871
86	0,802	0,919	0,872	0,826	0,535	0,884	0,756	0,616	0,826	0,767	0,884	0,826	0,907	0,465	0,581	0,942	0,872
87	0,805	0,92	0,874	0,828	0,529	0,885	0,747	0,621	0,828	0,759	0,885	0,828	0,908	0,471	0,586	0,943	0,874
88	0,807	0,921	0,875	0,83	0,523	0,886	0,75	0,614	0,83	0,761	0,875	0,83	0,909	0,466	0,591	0,943	0,875
89	0,809	0,921	0,876	0,832	0,528	0,876	0,753	0,618	0,832	0,764	0,876	0,832	0,91	0,472	0,596	0,944	0,876
90	0,811	0,922	0,878	0,833	0,522	0,878	0,744	0,622	0,833	0,756	0,878	0,833	0,911	0,478	0,6	0,944	0,878
91	0,813	0,923	0,879	0,835	0,517	0,879	0,747	0,626	0,835	0,758	0,879	0,835	0,912	0,473	0,593	0,945	0,879
92	0,815	0,924	0,88	0,837	0,511	0,88	0,739	0,62	0,837	0,761	0,88	0,837	0,913	0,467	0,587	0,946	0,88
93	0,817	0,925	0,882	0,839	0,505	0,882	0,731	0,624	0,839	0,753	0,882	0,839	0,914	0,462	0,591	0,946	0,882
94	0,809	0,926	0,883	0,84	0,5	0,883	0,734	0,628	0,84	0,745	0,883	0,84	0,915	0,457	0,585	0,947	0,883
95	0,811	0,926	0,884	0,842	0,495	0,884	0,737	0,632	0,842	0,747	0,874	0,842	0,916	0,453	0,579	0,947	0,884
96	0,813	0,927	0,885	0,844	0,49	0,885	0,729	0,625	0,844	0,75	0,875	0,844	0,917	0,448	0,583	0,948	0,885
97	0,814	0,928	0,887	0,845	0,495	0,887	0,732	0,629	0,845	0,753	0,876	0,845	0,918	0,443	0,577	0,949	0,887
98	0,816	0,929	0,888	0,847	0,49	0,888	0,735	0,622	0,847	0,755	0,878	0,847	0,918	0,439	0,582	0,949	0,888
99	0,818	0,929	0,889	0,849	0,495	0,889	0,737	0,626	0,849	0,758	0,879	0,849	0,919	0,444	0,586	0,95	0,889
100	0,82	0,93	0,89	0,85	0,49	0,88	0,74	0,63	0,85	0,76	0,88	0,85	0,92	0,45	0,59	0,95	0,89
101	0,822	0,931	0,891	0,852	0,495	0,881	0,743	0,634	0,852	0,753	0,881	0,852	0,921	0,455	0,594	0,951	0,891
102	0,824	0,931	0,892	0,853	0,49	0,882	0,745	0,637	0,853	0,755	0,873	0,853	0,922	0,461	0,598	0,951	0,892
103	0,825	0,932	0,893	0,854	0,485	0,884	0,748	0,641	0,854	0,748	0,864	0,854	0,922	0,456	0,592	0,952	0,893
Means 4. Quartil	0,809	0,923	0,878	0,834	0,517	0,882	0,744	0,618	0,837	0,758	0,881	0,839	0,912	0,46	0,584	0,945	0,878

The table shows the relative coherence judgements for all participants of group 3.

	Acht	Indien	Post	Tulpe	Mais	Belgien	Italien	Telefon	Tasche	Affe	Torte	Platz	Sand	Kuchen	Boot	Zebra	Bach	Reifen
1	1	1	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0
2	1	1	0,5	1	1	1	1	0,5	0	1	0,5	0	0	1	1	0	0,5	0
3	1	1	0,333	0,667	0,667	0,667	0,667	0,667	0	1	0,333	0,333	0	1	1	0	0,333	0,333
4	0,75	0,75	0,25	0,75	0,5	0,75	0,75	0,75	0	1	0,5	0,25	0	0,75	1	0	0,5	0,5
5	0,6	0,6	0,4	0,6	0,6	0,8	0,8	0,8	0	1	0,6	0,2	0	0,8	0,8	0	0,6	0,4
6	0,5	0,667	0,5	0,667	0,667	0,667	0,833	0,667	0	0,833	0,667	0,333	0	0,667	0,667	0,167	0,667	0,333
7	0,571	0,571	0,571	0,714	0,714	0,714	0,857	0,714	0	0,857	0,714	0,429	0	0,714	0,714	0,143	0,571	0,429
8	0,5	0,625	0,625	0,75	0,75	0,625	0,75	0,75	0,125	0,875	0,75	0,5	0	0,75	0,75	0,25	0,625	0,375
9	0,556	0,556	0,667	0,667	0,778	0,667	0,778	0,778	0,111	0,889	0,667	0,556	0,111	0,667	0,778	0,333	0,556	0,444
10	0,5	0,6	0,7	0,7	0,8	0,6	0,8	0,7	0,1	0,9	0,6	0,6	0,1	0,7	0,7	0,4	0,5	0,5
11	0,455	0,636	0,727	0,727	0,818	0,636	0,818	0,727	0,182	0,909	0,546	0,636	0,091	0,636	0,727	0,455	0,546	0,546
12	0,417	0,583	0,75	0,667	0,833	0,583	0,833	0,75	0,25	0,917	0,583	0,667	0,083	0,583	0,75	0,5	0,5	0,583
13	0,462	0,615	0,769	0,615	0,846	0,615	0,846	0,769	0,231	0,923	0,615	0,692	0,077	0,615	0,769	0,539	0,539	0,539
14	0,5	0,643	0,786	0,643	0,786	0,643	0,786	0,786	0,286	0,929	0,643	0,714	0,071	0,571	0,786	0,5	0,571	0,571
15	0,533	0,667	0,8	0,6	0,8	0,667	0,733	0,8	0,267	0,933	0,6	0,733	0,067	0,6	0,8	0,533	0,6	0,533
16	0,563	0,688	0,813	0,625	0,813	0,688	0,75	0,813	0,25	0,938	0,563	0,688	0,063	0,625	0,75	0,5	0,625	0,563
17	0,529	0,706	0,765	0,647	0,824	0,647	0,706	0,765	0,294	0,882	0,588	0,647	0,059	0,647	0,706	0,529	0,588	0,529
18	0,556	0,722	0,778	0,667	0,833	0,667	0,722	0,778	0,278	0,889	0,611	0,667	0,111	0,667	0,722	0,556	0,556	0,556
19	0,579	0,737	0,79	0,684	0,842	0,684	0,684	0,737	0,316	0,842	0,632	0,632	0,105	0,632	0,684	0,579	0,579	0,579
20	0,55	0,75	0,8	0,7	0,85	0,7	0,7	0,7	0,35	0,85	0,65	0,65	0,15	0,6	0,65	0,55	0,6	0,6
21	0,571	0,714	0,81	0,714	0,857	0,714	0,667	0,714	0,333	0,857	0,667	0,667	0,191	0,619	0,667	0,571	0,571	0,619
22	0,546	0,727	0,773	0,727	0,864	0,682	0,636	0,682	0,364	0,818	0,636	0,636	0,182	0,636	0,636	0,546	0,591	0,591
23	0,522	0,739	0,739	0,739	0,87	0,696	0,652	0,696	0,391	0,826	0,652	0,609	0,217	0,609	0,652	0,522	0,609	0,609
24	0,542	0,708	0,75	0,708	0,875	0,708	0,625	0,708	0,375	0,833	0,625	0,625	0,25	0,625	0,667	0,542	0,583	0,625
25	0,52	0,72	0,76	0,72	0,88	0,72	0,64	0,72	0,4	0,84	0,6	0,6	0,24	0,6	0,68	0,52	0,56	0,64
26	0,5	0,692	0,769	0,731	0,885	0,731	0,654	0,692	0,423	0,846	0,615	0,615	0,269	0,615	0,654	0,539	0,577	0,615
Mean 1. Quartil	0,589	0,708	0,689	0,709	0,806	0,703	0,757	0,699	0,205	0,899	0,583	0,526	0,094	0,69	0,758	0,376	0,54	0,485
27	0,482	0,704	0,778	0,704	0,889	0,741	0,667	0,704	0,444	0,852	0,63	0,63	0,296	0,63	0,667	0,556	0,593	0,63
28	0,464	0,679	0,786	0,714	0,893	0,714	0,679	0,679	0,464	0,857	0,607	0,643	0,286	0,643	0,679	0,571	0,607	0,643
29	0,448	0,69	0,793	0,724	0,897	0,724	0,69	0,655	0,448	0,862	0,621	0,655	0,276	0,655	0,655	0,552	0,621	0,655
30	0,467	0,667	0,8	0,733	0,9	0,733	0,7	0,667	0,433	0,867	0,633	0,667	0,267	0,667	0,667	0,567	0,6	0,667
31	0,484	0,645	0,807	0,742	0,903	0,71	0,71	0,677	0,419	0,871	0,645	0,677	0,29	0,677	0,677	0,548	0,613	0,677
32	0,469	0,625	0,813	0,75	0,906	0,688	0,719	0,656	0,438	0,875	0,625	0,688	0,281	0,688	0,656	0,531	0,594	0,656
33	0,455	0,636	0,818	0,727	0,909	0,697	0,697	0,667	0,424	0,879	0,606	0,667	0,273	0,667	0,667	0,515	0,576	0,636
34	0,471	0,647	0,824	0,735	0,912	0,706	0,706	0,677	0,412	0,882	0,618	0,677	0,265	0,677	0,677	0,529	0,559	0,647
35	0,486	0,629	0,829	0,714	0,914	0,714	0,714	0,657	0,429	0,886	0,6	0,686	0,286	0,686	0,686	0,543	0,571	0,657
36	0,5	0,639	0,833	0,722	0,917	0,694	0,722	0,667	0,444	0,889	0,611	0,694	0,306	0,694	0,667	0,556	0,583	0,639
37	0,514	0,649	0,838	0,703	0,919	0,703	0,73	0,676	0,46	0,892	0,595	0,703	0,324	0,676	0,649	0,568	0,568	0,649
38	0,526	0,658	0,842	0,711	0,921	0,711	0,737	0,684	0,474	0,895	0,605	0,711	0,342	0,684	0,632	0,579	0,553	0,658
39	0,539	0,667	0,846	0,718	0,923	0,718	0,744	0,692	0,487	0,897	0,615	0,718	0,359	0,692	0,641	0,59	0,564	0,667
40	0,55	0,675	0,85	0,7	0,925	0,725	0,75	0,7	0,5	0,9	0,6	0,725	0,375	0,675	0,65	0,6	0,575	0,675
41	0,561	0,683	0,854	0,707	0,927	0,732	0,756	0,707	0,488	0,902	0,61	0,732	0,39	0,683	0,634	0,61	0,585	0,683
42	0,571	0,691	0,857	0,714	0,929	0,738	0,762	0,714	0,5	0,905	0,595	0,738	0,381	0,691	0,619	0,619	0,571	0,691
43	0,581	0,698	0,861	0,721	0,93	0,744	0,767	0,721	0,512	0,907	0,581	0,744	0,395	0,674	0,628	0,628	0,581	0,698
44	0,568	0,682	0,864	0,727	0,932	0,727	0,773	0,727	0,523	0,909	0,591	0,727	0,409	0,659	0,636	0,614	0,568	0,705
45	0,556	0,689	0,867	0,733	0,933	0,711	0,778	0,733	0,533	0,911	0,578	0,711	0,422	0,644	0,644	0,6	0,556	0,689
46	0,565	0,674	0,87	0,739	0,935	0,717	0,783	0,739	0,522	0,913	0,587	0,717	0,435	0,652	0,652	0,609	0,565	0,674
47	0,553	0,66	0,872	0,723	0,936	0,723	0,787	0,745	0,532	0,915	0,596	0,723	0,447	0,66	0,66	0,596	0,575	0,681
48	0,563	0,667	0,875	0,729	0,938	0,729	0,792	0,75	0,521	0,917	0,604	0,729	0,458	0,667	0,667	0,604	0,583	0,688
49	0,571	0,674	0,878	0,735	0,939	0,735	0,796	0,735	0,531	0,918	0,612	0,714	0,469	0,653	0,674	0,592	0,592	0,694
50	0,58	0,66	0,88	0,72	0,94	0,74	0,8	0,74	0,52	0,92	0,62	0,72	0,48	0,64	0,68	0,6	0,6	0,7
51	0,569	0,667	0,882	0,706	0,941	0,745	0,804	0,745	0,529	0,922	0,628	0,706	0,49	0,628	0,686	0,608	0,588	0,706
52	0,577	0,673	0,885	0,712	0,942	0,75	0,808	0,75	0,519	0,923	0,635	0,712	0,5	0,635	0,692	0,615	0,596	0,712
Means 2. Quartil	0,526	0,666	0,842	0,722	0,921	0,722	0,745	0,702	0,481	0,895	0,61	0,7	0,365	0,665	0,659	0,581	0,582	0,672
53	0,585	0,679	0,887	0,698	0,943	0,755	0,811	0,755	0,509	0,925	0,642	0,717	0,509	0,642	0,698	0,623	0,604	0,717
54	0,593	0,667	0,889	0,704	0,944	0,759	0,815	0,759	0,5	0,926	0,648	0,722	0,519	0,648	0,704	0,63	0,611	0,722
55	0,582	0,673	0,891	0,709	0,946	0,764	0,818	0,764	0,509	0,927	0,655	0,727	0,527	0,655	0,691	0,636	0,6	0,727
56	0,589	0,679	0,893	0,714	0,946	0,768	0,821	0,768	0,5	0,929	0,661	0,732	0,536	0,661	0,696	0,643	0,607	0,732
57	0,597	0,684	0,895	0,719	0,947	0,772	0,825	0,772	0,491	0,93	0,667	0,737	0,544	0,649	0,702	0,649	0,614	0,737
58	0,603	0,69	0,897	0,707	0,948	0,776	0,828	0,776	0,483	0,931	0,672	0,741	0,552	0,655	0,707	0,655	0,621	0,741
59	0,593	0,678	0,898	0,712	0,949	0,763	0,831	0,78	0,492	0,932	0,678	0,729	0,542	0,644	0,712	0,644	0,61	0,746
60	0,583	0,683	0,9	0,7	0,95	0,75	0,833	0,783	0,5	0,933	0,683	0,717	0,533	0,633	0,717	0,633	0,6	0,75
61	0,574	0,689	0,902	0,705	0,951	0,738	0,836	0,771	0,492	0,918	0,689	0,705	0,541	0,623	0,705	0,639	0,607	0,738
62	0,581	0,677	0,903	0,694	0,952	0,742	0,823	0,774	0,484	0,919	0,694	0,71	0,548	0,629	0,71	0,645	0,613	0,742
63	0,587	0,683	0,905	0,698	0,952	0,746	0,825	0,778	0,476	0,921	0,683	0,714	0,54	0,619	0,714	0,651	0,619	0,746
64	0,594	0,688	0,906	0,688	0,953	0,75	0,828	0,781	0,469	0,922	0,688	0,719	0,547	0,625	0,719	0,656	0,625	0,75
65	0,6	0,677	0,908	0,692	0,954	0,754	0,831	0,785	0,462	0,923	0,692	0,723	0,539	0,615	0,723	0,662	0,631	0,754
66	0,606	0,682	0,909	0,682	0,955	0,742	0,833	0,788	0,47	0,924	0,697	0,727	0,546	0,606	0,727	0,667	0,636	0,758
67	0,612	0,687	0,91	0,687	0,955	0,746	0,836	0,791	0,478	0,925	0,687	0,731	0,537	0,612	0,716	0,672	0,627	0,761
68	0,603	0,677	0,912	0,677	0,956	0,75	0,838	0,794	0,485	0,927	0,677	0,721	0,544	0,618	0,721	0,677	0,618	0,765
69	0,609	0,667	0,913	0,667	0,957	0,754	0,841	0,797	0,493	0,928	0,681	0,725	0,551	0,623	0,725	0,681	0,623	0,768
70	0,614	0,657	0,914	0,657	0,957	0,757	0,843	0,8	0,5	0,929	0,671	0,729	0,557	0,629	0,729	0,686	0,629	0,771
71	0,606	0,648	0,916	0,662	0,958	0,761	0,845	0,803	0,493	0,93	0,662	0,732	0,549	0,62	0,718	0,69	0,62	0,775
72	0,597	0,653	0,917	0,667	0,958	0,75	0,847	0,806	0,5	0,931	0,667	0,736	0,556	0,611	0,722	0,694	0,625	0,764
73	0,589	0,644	0,918	0,658	0,959	0,753	0,836	0,808	0,507	0,932	0,658	0,74	0,562	0,603	0,726	0,699	0,63	0,767
74	0,581	0,649	0,919	0,649	0,96	0,743	0,838	0,811	0,514	0,932	0,649	0,743	0,554	0,595	0,73	0,689	0,622	0,77
75	0,573	0,64	0,92	0,653	0,96	0,747	0,84	0,813	0,52	0,933	0,653	0,747	0,56	0,587	0,733	0,693	0,627	0,76
76	0,566	0,645	0,921	0,645	0,961	0,737	0,842	0,816	0,526	0,934	0,658	0,75	0,566	0,579	0,737	0,697	0,618	0,763
77	0,571	0,649	0,922	0,649	0,961	0,74	0,831	0,818	0,52	0,935	0,662	0,753	0,571	0,584	0,74	0,701	0,61	0,766
78	0,564	0,654	0,923	0,641	0,962	0,731	0,821	0,821	0,526	0,936	0,667	0,756	0,564	0,577	0,744	0,692	0,615	0,769
Means 3. Quartil	0,59	0,669	0,907	0,682	0,954	0,752	0,831	0,789	0,496	0,928	0,671	0,73	0,546	0,621	0,718	0,666	0,618	0,752
79	0,557	0,658	0,924	0,633	0,962	0,734	0,823	0,81	0,532	0,937	0,658	0,76	0,57	0,57	0,747	0,684	0,62	0,76
80	0,55	0,663	0,925	0,638	0,963	0,738	0,825	0,813	0,538	0,938	0,663	0,75	0,575	0,563	0,75	0,675	0,625	0,763
81	0,556	0,654	0,926	0,642	0,963	0,741	0,827	0,815	0,531	0,938	0,654	0,753	0,568	0,568	0,753	0,679	0,617	0,765
82	0,549	0,646	0,927	0,646	0,963	0,744	0,829	0,817	0,537	0,939	0,659	0,756	0,561	0,573	0,756	0,683	0,61	0,768
83	0,542	0,639	0,928	0,639	0,964	0,747	0,831	0,807	0,53	0,94	0,651	0,759	0,554	0,578	0,759	0,687	0,602	0,771
84	0,548	0,643	0,929	0,643	0,964	0,75	0,833	0,798	0,536	0,941	0,655	0,762	0,56	0,571	0,762	0,679	0,607	0,762
85	0,553	0,635	0,929	0,647	0,965	0,753	0,835	0,8	0,529	0,941	0,647	0,765	0,553	0,577	0,765	0,682	0,6	0,765
86	0,547	0,64	0,93	0,64	0,965	0,744	0,837	0,802	0,535	0,942	0,64	0,767	0,547	0,57	0,767	0,674	0,605	0,767
87	0,552	0,644	0,931	0,632	0,966	0,747	0,839	0,805	0,54	0,943	0,644	0,759	0,54	0,563	0,77	0,678	0,598	0,77
88	0,546	0,648	0,932	0,636	0,966	0,75	0,841	0,807	0,534	0,943	0,636	0,761	0,546	0,568	0,773	0,682	0,602	0,773
89	0,551	0,652	0,933	0,64	0,966	0,742	0,843	0,809	0,539	0,944	0,629	0,753	0,539	0,573	0,775	0,674	0,607	0,764
90	0,556	0,656	0,933	0,644	0,967	0,744	0,844	0,811	0,533	0,944	0,633	0,756	0,544	0,578	0,778	0,678	0,611	0,767
91	0,55	0,659	0,934	0,637	0,967	0,747	0,835	0,813	0,539	0,945	0,626	0,747	0,539	0,571	0,78	0,67	0,615	0,769
92	0,554	0,663	0,935	0,63	0,967	0,739	0,837	0,815	0,544	0,946	0,62	0,75	0,544	0,565	0,783	0,663	0,62	0,761
93	0,559	0,667	0,936	0,634	0,968	0,742	0,839	0,817	0,538	0,946	0,624	0,753	0,548	0,57	0,785	0,667	0,613	0,763
94	0,564	0,67	0,936	0,638	0,968	0,745	0,84	0,819	0,532	0,947	0,628	0,755	0,553	0,575	0,777	0,67	0,606	0,766
95	0,568	0,663	0,937	0,642	0,968	0,747	0,842	0,821	0,537	0,947	0,632	0,758	0,558	0,579	0,779	0,674	0,611	0,768
96	0,563	0,667	0,938	0,635	0,969	0,74	0,844	0,823	0,542	0,948	0,635	0,76	0,552	0,573	0,781	0,667	0,615	0,771
97	0,557	0,67	0,938	0,639	0,969	0,742	0,835	0,814	0,536	0,949	0,629	0,763	0,557	0,577	0,784	0,67	0,619	0,773
98	0,551	0,674	0,939	0,643	0,969	0,745	0,837	0,816	0,541	0,949	0,633	0,755	0,551	0,571	0,786	0,663	0,612	0,776
99	0,546	0,677	0,939	0,647	0,97	0,748	0,838	0,818	0,546	0,95	0,626	0,758	0,556	0,566	0,778	0,667	0,616	0,778
100	0,54	0,68	0,94	0,64	0,97	0,75	0,84	0,82	0,55	0,95	0,63	0,76	0,55	0,56	0,78	0,67	0,61	0,78
101	0,545	0,673	0,941	0,634	0,97	0,753	0,842	0,822	0,545	0,951	0,634	0,762	0,555	0,555	0,782	0,673	0,604	0,782
102	0,549	0,667	0,941	0,637	0,971	0,755	0,843	0,824	0,549	0,951	0,637	0,765	0,559	0,549	0,784	0,677	0,608	0,784
103	0,544	0,66	0,942	0,641	0,971	0,757	0,835	0,825	0,553	0,952	0,641	0,767	0,563	0,544	0,786	0,68	0,612	0,786
Means 4. Quartil	0,552	0,659	0,934	0,639	0,967	0,746	0,837	0,814	0,539	0,945	0,638	0,758	0,554	0,568	0,773	0,675	0,611	0,77

The table shows the relative coherence judgements for all participants of group 4.

	wedel751	wedel714	wedel804	wedel524	wedel673	wedel480	wedel723	wedel836	wedel563	wedel630	wedel692	wedel579	wedel603	wedel862
1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
2	1	1	0,5	1	1	1	0,5	1	1	0,5	1	1	1	1
3	1	0,667	0,667	1	0,667	0,667	0,333	1	1	0,667	0,667	0,667	1	0,667
4	1	0,75	0,75	0,75	0,5	0,75	0,25	1	1	0,5	0,5	0,5	0,75	0,5
5	1	0,6	0,8	0,8	0,4	0,8	0,4	0,8	1	0,6	0,4	0,6	0,6	0,6
6	1	0,667	0,667	0,833	0,5	0,833	0,5	0,833	1	0,5	0,333	0,667	0,5	0,667
7	1	0,714	0,714	0,714	0,429	0,857	0,571	0,857	1	0,571	0,429	0,714	0,429	0,714
8	1	0,625	0,625	0,75	0,5	0,875	0,625	0,75	1	0,625	0,5	0,75	0,5	0,75
9	1	0,667	0,667	0,778	0,556	0,889	0,667	0,778	1	0,556	0,444	0,778	0,556	0,778
10	0,9	0,7	0,7	0,8	0,6	0,8	0,6	0,8	1	0,6	0,5	0,7	0,5	0,7
11	0,909	0,727	0,636	0,727	0,546	0,727	0,546	0,818	1	0,636	0,546	0,636	0,546	0,636
12	0,917	0,75	0,583	0,75	0,5	0,75	0,583	0,833	1	0,667	0,583	0,583	0,583	0,667
13	0,923	0,769	0,615	0,769	0,462	0,769	0,615	0,846	1	0,692	0,615	0,615	0,615	0,692
14	0,929	0,786	0,571	0,786	0,429	0,786	0,643	0,857	1	0,643	0,643	0,571	0,643	0,714
15	0,933	0,8	0,6	0,8	0,467	0,8	0,667	0,867	1	0,667	0,667	0,6	0,667	0,733
16	0,938	0,813	0,625	0,813	0,438	0,813	0,688	0,875	1	0,625	0,625	0,625	0,688	0,75
17	0,882	0,765	0,588	0,765	0,471	0,765	0,706	0,824	1	0,588	0,588	0,588	0,647	0,706
18	0,889	0,778	0,611	0,778	0,5	0,778	0,722	0,833	1	0,611	0,611	0,611	0,667	0,722
19	0,895	0,737	0,579	0,79	0,526	0,79	0,737	0,842	1	0,579	0,579	0,632	0,684	0,737
20	0,9	0,75	0,6	0,8	0,5	0,8	0,7	0,8	1	0,6	0,6	0,6	0,65	0,7
21	0,905	0,714	0,619	0,81	0,524	0,81	0,714	0,81	1	0,619	0,619	0,619	0,667	0,714
22	0,864	0,682	0,591	0,773	0,5	0,773	0,727	0,773	1	0,591	0,591	0,591	0,636	0,682
23	0,87	0,696	0,609	0,739	0,522	0,783	0,696	0,739	1	0,565	0,565	0,609	0,609	0,652
24	0,875	0,708	0,583	0,75	0,542	0,792	0,708	0,75	1	0,583	0,583	0,625	0,625	0,667
25	0,88	0,72	0,6	0,72	0,56	0,8	0,72	0,76	1	0,56	0,56	0,64	0,64	0,68
26	0,885	0,731	0,577	0,731	0,577	0,808	0,692	0,769	1	0,577	0,577	0,654	0,654	0,654
Means 1. Quartil	0,934	0,743	0,641	0,797	0,547	0,808	0,627	0,839	1	0,574	0,589	0,661	0,656	0,711
27	0,889	0,741	0,593	0,741	0,593	0,815	0,704	0,778	1	0,593	0,593	0,667	0,667	0,667
28	0,893	0,714	0,607	0,714	0,607	0,821	0,679	0,786	1	0,607	0,607	0,643	0,679	0,643
29	0,897	0,724	0,621	0,724	0,621	0,828	0,69	0,793	1	0,621	0,621	0,655	0,655	0,655
30	0,9	0,733	0,633	0,733	0,633	0,833	0,7	0,8	1	0,6	0,633	0,667	0,667	0,667
31	0,903	0,742	0,645	0,71	0,645	0,839	0,71	0,807	1	0,613	0,645	0,677	0,677	0,677
32	0,906	0,75	0,625	0,688	0,656	0,844	0,688	0,813	1	0,625	0,656	0,688	0,688	0,656
33	0,909	0,758	0,636	0,667	0,667	0,849	0,667	0,788	1	0,636	0,636	0,667	0,697	0,636
34	0,912	0,765	0,647	0,647	0,647	0,853	0,677	0,794	1	0,618	0,647	0,677	0,706	0,647
35	0,914	0,771	0,657	0,657	0,657	0,857	0,686	0,8	1	0,629	0,657	0,686	0,714	0,657
36	0,917	0,75	0,667	0,667	0,667	0,861	0,694	0,806	1	0,639	0,667	0,694	0,722	0,667
37	0,919	0,757	0,649	0,676	0,676	0,865	0,676	0,811	1	0,649	0,676	0,703	0,73	0,676
38	0,921	0,763	0,658	0,684	0,658	0,868	0,684	0,816	1	0,632	0,684	0,711	0,737	0,684
39	0,923	0,769	0,667	0,667	0,641	0,872	0,692	0,821	1	0,615	0,692	0,718	0,744	0,692
40	0,925	0,775	0,675	0,65	0,65	0,875	0,7	0,8	1	0,6	0,7	0,725	0,75	0,7
41	0,927	0,781	0,683	0,659	0,634	0,878	0,707	0,805	1	0,61	0,707	0,732	0,756	0,707
42	0,929	0,786	0,667	0,667	0,643	0,881	0,691	0,81	1	0,619	0,714	0,738	0,762	0,714
43	0,93	0,791	0,674	0,674	0,628	0,884	0,698	0,791	1	0,605	0,721	0,721	0,767	0,721
44	0,932	0,796	0,659	0,682	0,614	0,886	0,682	0,796	1	0,614	0,727	0,705	0,75	0,705
45	0,933	0,8	0,667	0,689	0,622	0,889	0,689	0,8	1	0,622	0,711	0,711	0,756	0,711
46	0,935	0,804	0,674	0,696	0,63	0,891	0,696	0,804	1	0,609	0,717	0,717	0,761	0,717
47	0,936	0,809	0,66	0,702	0,638	0,894	0,681	0,787	1	0,596	0,702	0,723	0,745	0,702
48	0,938	0,813	0,667	0,688	0,625	0,896	0,688	0,792	1	0,583	0,708	0,729	0,75	0,708
49	0,939	0,816	0,674	0,694	0,612	0,898	0,674	0,796	1	0,592	0,714	0,735	0,735	0,694
50	0,94	0,82	0,68	0,7	0,6	0,9	0,68	0,8	1	0,6	0,72	0,74	0,74	0,7
51	0,941	0,824	0,667	0,706	0,608	0,902	0,686	0,804	1	0,608	0,726	0,745	0,745	0,706
52	0,942	0,827	0,673	0,712	0,596	0,904	0,692	0,808	1	0,615	0,731	0,75	0,75	0,712
Means 2. Quartil	0,921	0,776	0,655	0,688	0,633	0,869	0,689	0,8	1	0,613	0,681	0,705	0,725	0,685
53	0,943	0,83	0,679	0,717	0,585	0,906	0,698	0,811	1	0,623	0,736	0,755	0,755	0,717
54	0,944	0,833	0,685	0,722	0,574	0,907	0,704	0,815	1	0,63	0,741	0,759	0,759	0,722
55	0,946	0,836	0,691	0,727	0,582	0,909	0,691	0,818	1	0,636	0,746	0,764	0,764	0,727
56	0,946	0,821	0,696	0,732	0,589	0,911	0,696	0,821	1	0,643	0,75	0,768	0,768	0,732
57	0,947	0,825	0,684	0,737	0,597	0,912	0,702	0,807	1	0,649	0,737	0,772	0,754	0,737
58	0,948	0,828	0,69	0,741	0,586	0,914	0,707	0,81	1	0,655	0,741	0,776	0,759	0,741
59	0,949	0,831	0,678	0,746	0,593	0,915	0,695	0,797	1	0,661	0,746	0,763	0,763	0,746
60	0,95	0,833	0,683	0,75	0,583	0,917	0,7	0,8	1	0,667	0,75	0,75	0,767	0,75
61	0,934	0,82	0,672	0,738	0,59	0,902	0,705	0,803	1	0,672	0,738	0,754	0,771	0,754
62	0,936	0,823	0,677	0,742	0,597	0,903	0,71	0,807	1	0,677	0,742	0,758	0,774	0,758
63	0,937	0,825	0,683	0,746	0,603	0,905	0,714	0,81	1	0,683	0,746	0,762	0,778	0,762
64	0,938	0,828	0,672	0,75	0,594	0,906	0,719	0,813	1	0,688	0,75	0,766	0,766	0,766
65	0,939	0,831	0,662	0,739	0,6	0,908	0,723	0,8	1	0,677	0,754	0,769	0,769	0,769
66	0,939	0,833	0,667	0,742	0,606	0,909	0,727	0,803	1	0,682	0,742	0,773	0,773	0,773
67	0,94	0,836	0,672	0,746	0,612	0,91	0,731	0,806	1	0,687	0,746	0,776	0,776	0,776
68	0,941	0,838	0,662	0,75	0,603	0,912	0,735	0,809	1	0,691	0,75	0,779	0,779	0,779
69	0,942	0,841	0,667	0,754	0,609	0,913	0,739	0,812	1	0,696	0,754	0,783	0,783	0,783
70	0,943	0,843	0,657	0,757	0,614	0,914	0,743	0,814	1	0,7	0,757	0,786	0,786	0,786
71	0,944	0,845	0,662	0,761	0,606	0,916	0,732	0,817	1	0,704	0,761	0,789	0,789	0,789
72	0,944	0,847	0,653	0,764	0,611	0,917	0,736	0,819	1	0,708	0,75	0,792	0,792	0,792
73	0,945	0,849	0,644	0,767	0,616	0,918	0,74	0,822	1	0,712	0,753	0,795	0,795	0,795
74	0,946	0,851	0,649	0,77	0,622	0,919	0,73	0,824	1	0,703	0,743	0,797	0,797	0,784
75	0,947	0,853	0,64	0,773	0,627	0,92	0,733	0,827	1	0,707	0,733	0,787	0,8	0,787
76	0,947	0,855	0,645	0,776	0,618	0,921	0,724	0,829	1	0,711	0,724	0,79	0,803	0,79
77	0,948	0,857	0,649	0,779	0,623	0,922	0,727	0,831	1	0,714	0,727	0,792	0,805	0,792
78	0,949	0,859	0,654	0,782	0,615	0,923	0,731	0,833	1	0,705	0,731	0,782	0,795	0,795
Means 3. Quartil	0,944	0,837	0,668	0,75	0,602	0,913	0,719	0,814	1	0,68	0,744	0,774	0,778	0,765
79	0,949	0,848	0,658	0,785	0,62	0,911	0,722	0,835	1	0,709	0,734	0,772	0,798	0,798
80	0,95	0,85	0,65	0,775	0,613	0,913	0,725	0,838	1	0,713	0,725	0,763	0,788	0,8
81	0,938	0,852	0,642	0,778	0,617	0,914	0,728	0,84	1	0,716	0,728	0,765	0,79	0,803
82	0,939	0,854	0,646	0,781	0,622	0,915	0,732	0,842	1	0,72	0,732	0,768	0,793	0,805
83	0,94	0,855	0,639	0,783	0,615	0,916	0,735	0,843	1	0,723	0,735	0,771	0,795	0,807
84	0,941	0,857	0,643	0,786	0,619	0,917	0,726	0,845	1	0,726	0,738	0,774	0,798	0,81
85	0,941	0,859	0,635	0,788	0,624	0,918	0,729	0,847	1	0,729	0,741	0,777	0,8	0,812
86	0,942	0,861	0,64	0,791	0,616	0,919	0,721	0,849	1	0,721	0,733	0,779	0,802	0,802
87	0,943	0,862	0,632	0,793	0,609	0,92	0,724	0,851	1	0,713	0,736	0,782	0,805	0,805
88	0,943	0,864	0,625	0,796	0,602	0,921	0,716	0,852	1	0,716	0,739	0,784	0,807	0,807
89	0,944	0,865	0,629	0,787	0,607	0,921	0,719	0,854	1	0,719	0,742	0,787	0,809	0,809
90	0,944	0,867	0,633	0,789	0,6	0,922	0,722	0,856	1	0,722	0,744	0,789	0,811	0,811
91	0,945	0,868	0,637	0,78	0,604	0,923	0,714	0,857	1	0,725	0,736	0,78	0,813	0,813
92	0,946	0,87	0,63	0,783	0,609	0,924	0,707	0,859	1	0,728	0,739	0,772	0,804	0,815
93	0,946	0,871	0,634	0,774	0,613	0,925	0,71	0,86	1	0,731	0,742	0,774	0,807	0,817
94	0,947	0,872	0,628	0,766	0,617	0,926	0,713	0,862	1	0,734	0,745	0,777	0,809	0,819
95	0,947	0,874	0,621	0,768	0,621	0,926	0,716	0,863	1	0,737	0,747	0,779	0,811	0,821
96	0,948	0,875	0,615	0,771	0,625	0,927	0,719	0,865	1	0,74	0,75	0,781	0,813	0,823
97	0,949	0,876	0,619	0,773	0,629	0,928	0,711	0,866	1	0,742	0,753	0,784	0,814	0,825
98	0,949	0,878	0,622	0,765	0,622	0,929	0,704	0,867	1	0,745	0,745	0,776	0,806	0,827
99	0,95	0,879	0,616	0,768	0,626	0,929	0,697	0,869	1	0,748	0,748	0,768	0,798	0,828
100	0,95	0,88	0,62	0,76	0,62	0,93	0,7	0,87	1	0,75	0,74	0,77	0,8	0,83
101	0,951	0,881	0,624	0,762	0,614	0,931	0,703	0,871	1	0,753	0,733	0,772	0,802	0,832
102	0,951	0,882	0,618	0,765	0,608	0,931	0,706	0,873	1	0,755	0,735	0,775	0,804	0,833
103	0,952	0,884	0,612	0,767	0,612	0,932	0,709	0,874	1	0,757	0,738	0,777	0,806	0,835
Means 4. Quartil	0,946	0,867	0,631	0,777	0,615	0,923	0,716	0,856	1	0,731	0,739	0,776	0,803	0,815

The following tables show the complete results of the paired t-tests. Each t-test compares the mean values of two quartils.

t-Test: Paired Two Sample for Means

	1. Quartile	*2. Quartile*
Mean	0,66981018	0,71845368
Variance	0,02791586	0,01784974
Observations	71	71
Degrees of freedom (df)	70	
t Stat	-4,43618462	
P(T<=t) one-tail	1,6632E-05	
t Critical one-tail	2,38080746	

t-Test: Paired Two Sample for Means

	2. Quartile	*3. Quartile*
Mean	0,71845368	0,76881977
Variance	0,01784974	0,01613306
Observations	71	71
Degrees of freedom (df)	70	
t Stat	-10,644686	
P(T<=t) one-tail	1,395E-16	
t Critical one-tail	2,38080746	

t-Test: Paired Two Sample for Means

	3. Quartile	*4. Quartile*
Mean	0,76881977	0,78391234
Variance	0,01613306	0,01783453
Observations	71	71
Degrees of freedom (df)	70	
	-	
t Stat	4,66570024	
P(T<=t) one-tail	7,1732E-06	
t Critical one-tail	2,38080746	

Phase 3
H3.3. The following tables show the coherence judgements for participants that are A, B, C or D rated (according to the distance to the canonical class). Each cell contains a binary value (1=coherent judgement to her/his own classification; 0=incoherent judgement to her/his own classification).

ID	Apfel	Birne	Eule	Katze	Berlin	Bremen	Frosch	Farbe	Biber	Post	Mais	Reifen	wedel804	wedel524	wedel723	wedel563	wedel862	Zebra
Rating	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	invalid in phase #2
Distance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0
2	1	1	1	1	0	0	1	1	1	0	1	1	0	1	1	1	1	1
3	1	1	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0	1
4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
5	1	0	0	0	1	0	1	1	0	1	1	0	1	0	1	1	1	1
6	0	1	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1
7	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0
8	1	1	0	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
11	1	0	1	1	0	0	0	0	1	1	1	0	1	1	1	1	0	1
12	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
13	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1
14	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0
15	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
17	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
20	1	1	0	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0
21	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0
23	0	1	1	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0
24	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
25	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0
26	0	1	0	1	0	0	0	1	0	1	1	1	1	0	1	1	0	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	1	1	1	1	0	0	0	1	1	1	1	0	0	1	1	1	0	1
29	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
32	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0
33	1	0	0	1	1	1	0	1	0	1	1	0	0	0	1	1	0	0
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
35	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
36	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
37	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
38	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
39	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
40	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
42	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
43	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
44	0	1	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
45	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
46	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
47	1	1	1	1	1	0	0	1	1	1	1	0	1	0	1	1	0	0
48	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
49	0	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0	0
50	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
52	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
53	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
54	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
56	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
57	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
58	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
59	1	1	0	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0
60	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
61	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	1
62	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
63	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
64	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
65	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
67	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
68	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
69	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
71	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
72	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
73	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
74	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0	0
75	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
76	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	1
77	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
79	1	0	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0
80	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0
81	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
82	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
83	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
84	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0
85	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
86	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0	0
87	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
88	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
89	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
90	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
91	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0
92	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0
93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
94	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
95	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
96	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0
97	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
98	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0
99	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1
100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
102	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
103	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1

ID Rating	WolkeB	BananeB	LampeB	BlumeB	WasserB	KofferB	IgelB	BusB	GolfB	AmpelB	AchtB	AffeB	BootB	wedel751B	wedel714B	wedel480B	wedel692B	KerzeB	LichtB	FeuerB	BuchB	AulaB	TelefonB	wedel603B	SeifeB	SonneB	RegenB	HundB	GenfB	HausB	KranB
Distance	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3
1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1
2	1	0	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
3	1	1	0	0	1	1	1	1	0	1	1	0	0	1	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	1
4	1	1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1	0	0	0	1
5	0	1	0	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	0	1	1	1	1	1	0	1
6	0	0	0	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	0	1	1	1	1	0	0	1
7	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1
8	0	1	0	1	1	1	1	0	0	1	0	1	1	1	0	1	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	0	0	0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	1	1	0	1	0	1	1	1	0	0
11	1	1	1	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	1	1	1	1	1	0	1	1
12	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1
15	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0
16	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1
17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	0	1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
20	0	1	0	0	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	0	0	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
23	1	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1
24	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
25	1	1	1	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1
26	1	1	0	0	1	0	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
28	0	1	1	1	1	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1
30	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
31	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	0	1	0	1	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1
33	1	1	1	1	0	0	1	0	1	1	0	0	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
34	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
36	1	1	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
37	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1
38	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
39	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
41	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
42	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
43	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
44	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	0
45	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	0
46	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
47	1	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1
48	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
49	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0
50	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
51	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0
52	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
53	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
54	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
55	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
56	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0
57	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
58	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
59	1	1	0	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1
60	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1
61	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0
62	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
63	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
64	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
65	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1
67	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
68	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0
69	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
71	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
72	1	1	0	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1
73	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0
74	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1
75	1	1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	1
76	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1
77	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	1
79	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1	0	0	1	0	0	1
80	1	1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1	1	0	1	1
81	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
82	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
83	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0
84	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
85	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0
86	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1
87	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	0
88	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1
89	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
90	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
91	1	1	1	0	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1
92	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1
93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
94	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
95	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	1
96	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
97	1	1	1	1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
98	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1
99	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
100	1	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1	0	0	1	1	1	1	1
101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
102	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
103	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1

ID	Hamburg	Kiel	Indien	wedel836	Baum	Tiere	Belgien	Platz	Italien	Tasche
Rating	C	C	C	C	C	C	C	C	C	C
Distance	4	4	4	4	5	5	5	5	6	6
1	1	1	0	1	1	1	1	0	1	1
2	0	0	1	1	0	1	1	1	1	1
3	0	1	1	0	1	0	0	1	0	1
4	0	0	1	0	1	1	1	0	1	0
5	0	1	1	1	1	1	1	0	1	1
6	0	1	0	1	1	0	0	1	1	1
7	1	0	0	1	0	1	1	1	1	1
8	0	0	0	0	0	1	0	1	0	0
9	1	1	1	1	1	1	1	1	1	1
10	1	0	1	1	0	0	0	1	1	1
11	0	0	0	1	1	1	1	1	1	0
12	0	1	1	0	0	0	0	1	1	1
13	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	0	0	1	1	0	1
15	1	1	0	1	1	0	1	1	0	1
16	1	1	1	1	1	1	1	0	1	1
17	0	1	0	0	1	1	0	0	0	0
18	1	1	1	1	1	1	1	1	1	1
19	1	1	0	1	0	0	1	0	0	0
20	0	1	1	0	1	1	1	1	1	0
21	1	1	1	1	1	0	1	1	0	1
22	0	0	0	0	1	1	0	0	0	0
23	0	1	0	1	0	1	1	0	1	0
24	1	0	0	1	0	0	1	1	0	0
25	0	1	1	1	0	0	1	0	1	1
26	1	1	0	1	1	1	1	1	1	0
27	1	0	0	0	0	0	1	1	1	0
28	0	1	1	1	0	1	0	1	1	0
29	0	0	1	1	1	1	1	1	1	0
30	1	1	1	1	1	0	1	1	1	1
31	1	1	0	1	1	0	0	1	1	1
32	0	0	1	1	0	0	0	1	1	0
33	1	0	1	0	0	0	1	0	0	1
34	1	1	1	1	1	1	1	1	1	1
35	1	1	1	1	1	0	1	1	1	0
36	1	0	1	1	1	1	0	1	1	0
37	1	1	1	1	1	0	1	1	1	0
38	1	1	1	1	1	1	1	1	1	0
39	1	1	1	1	1	1	1	1	1	0
40	0	1	1	0	1	0	1	1	1	0
41	1	1	0	1	1	0	1	1	1	1
42	1	1	1	1	1	1	1	1	1	0
43	1	0	0	1	1	0	1	1	1	1
44	0	0	1	0	0	0	0	0	1	0
45	0	0	1	0	0	0	0	0	1	0
46	1	1	0	1	1	1	1	1	1	1
47	0	0	1	1	0	0	1	1	1	1
48	1	1	1	1	1	1	1	1	1	1
49	0	0	0	0	0	0	1	0	1	1
50	1	1	1	1	1	1	1	1	1	1
51	0	0	1	0	1	0	1	0	1	0
52	1	1	1	1	1	1	1	1	1	1
53	1	1	1	1	1	1	1	1	1	1
54	1	1	1	1	1	0	1	1	1	1
55	1	1	1	1	1	1	1	1	1	0
56	1	1	1	1	1	1	1	1	1	1
57	1	1	0	0	1	1	1	1	1	1
58	1	1	1	1	1	1	1	1	1	1
59	0	1	0	1	0	0	0	0	1	0
60	1	1	1	1	1	0	0	0	1	1
61	0	1	1	1	0	0	0	0	1	0
62	1	1	1	1	1	1	1	1	0	1
63	1	1	1	1	1	1	1	1	1	1
64	0	1	0	0	0	0	1	1	1	1
65	1	1	1	0	1	1	1	1	1	1
66	0	1	1	0	0	0	0	1	1	0
67	1	0	1	1	1	1	1	1	1	0
68	0	0	1	1	0	0	1	0	1	0
69	0	0	1	1	1	1	1	1	1	0
70	1	1	1	1	0	1	1	1	1	0
71	1	1	1	1	1	1	1	1	1	1
72	0	1	0	0	0	0	0	1	1	0
73	1	0	1	1	1	1	1	1	0	0
74	0	1	0	1	0	0	0	1	1	0
75	1	0	0	1	0	0	1	1	1	0
76	0	0	1	0	0	0	0	1	1	1
77	1	0	1	1	1	1	1	1	0	1
78	1	1	1	0	0	0	0	1	0	0
79	1	1	1	1	1	0	1	1	1	1
80	1	1	1	0	1	0	1	0	1	0
81	1	0	1	1	1	0	1	1	1	1
82	1	1	1	1	1	1	1	1	1	0
83	1	1	1	1	0	0	1	1	1	1
84	1	1	1	1	1	0	1	1	1	1
85	0	1	1	1	1	1	1	1	1	1
86	1	0	0	1	0	0	0	1	1	0
87	1	1	0	0	1	0	1	0	1	0
88	1	1	1	1	1	1	1	1	1	1
89	1	1	1	0	1	0	0	0	1	0
90	1	1	1	1	1	1	1	1	1	1
91	1	0	0	0	0	0	1	0	0	1
92	1	1	1	0	1	0	0	1	1	0
93	1	1	1	1	1	1	1	1	1	1
94	1	1	1	1	1	1	1	1	1	1
95	1	1	1	1	1	1	1	1	1	0
96	0	1	0	0	1	0	0	1	1	1
97	1	1	1	1	1	1	1	1	0	1
98	0	1	0	0	0	0	1	0	1	1
99	1	1	0	1	1	0	1	1	1	0
100	1	1	0	0	0	0	1	1	1	0
101	1	1	1	1	1	1	1	1	1	1
102	1	1	1	1	0	1	1	1	1	0
103	1	0	1	1	1	1	1	1	0	0

ID	Eisen	Elch	Sand	Tulpe	Kuchen	Bach	wedel673	wedel630
Rating	D	D	D	D	D	D	D	D
Distance	7	7	7	8	8	8	8	8
1	1	1	0	1	0	0	1	0
2	0	1	1	1	1	1	1	1
3	1	0	1	1	1	0	1	1
4	1	0	1	1	0	1	0	0
5	0	0	1	0	1	1	0	1
6	1	0	1	0	1	0	0	1
7	0	0	1	1	1	0	1	1
8	1	0	0	1	0	1	0	0
9	1	0	1	0	0	0	1	0
10	1	0	0	1	1	0	0	1
11	1	1	0	1	0	1	0	1
12	1	1	0	0	1	0	1	1
13	1	1	0	0	1	1	1	1
14	1	0	0	0	0	1	1	0
15	1	0	0	1	1	1	1	1
16	1	1	0	1	1	1	0	0
17	0	0	0	1	0	0	0	1
18	1	1	1	1	1	0	1	1
19	0	1	0	1	0	1	1	1
20	1	0	0	1	0	1	0	1
21	1	0	1	1	1	0	0	1
22	1	0	0	1	0	1	0	0
23	1	1	0	1	0	1	1	0
24	0	0	1	0	1	0	1	1
25	0	0	0	0	0	0	0	0
26	1	1	1	1	1	1	1	1
27	1	0	1	0	0	1	1	1
28	1	0	0	1	1	1	1	1
29	1	1	1	1	0	1	1	0
30	0	1	0	1	1	0	1	0
31	0	0	1	1	1	1	1	1
32	1	1	0	0	1	0	1	1
33	1	0	0	0	0	0	0	1
34	1	1	0	1	1	0	0	0
35	1	0	1	1	1	1	0	1
36	1	1	1	1	1	1	1	1
37	1	1	1	0	0	0	0	0
38	1	0	1	1	1	0	0	0
39	1	0	1	1	1	1	1	0
40	1	0	0	0	1	1	1	1
41	0	1	1	0	1	1	1	1
42	1	0	0	1	1	0	0	1
43	1	1	0	0	0	1	0	0
44	0	1	0	1	0	0	1	1
45	1	0	1	0	0	0	0	1
46	0	0	1	1	1	1	1	0
47	1	1	1	0	0	1	1	1
48	1	1	1	1	1	1	0	0
49	1	0	1	1	0	1	0	1
50	1	0	1	0	1	1	1	1
51	1	1	1	0	0	0	1	1
52	1	0	1	1	1	1	0	1
53	1	0	1	1	1	1	0	1
54	1	1	1	1	1	1	0	1
55	1	1	1	1	1	0	1	1
56	1	1	1	1	1	1	1	1
57	1	0	0	1	0	1	1	1
58	1	1	1	1	1	1	0	1
59	0	0	0	1	0	0	0	1
60	1	0	0	0	0	0	0	1
61	1	1	1	1	0	1	1	1
62	1	1	1	1	1	1	1	1
63	1	0	1	0	1	1	1	1
64	1	0	0	0	1	1	0	1
65	1	1	0	1	0	1	1	0
66	1	0	1	0	0	1	1	0
67	1	0	0	1	1	0	1	1
68	1	0	1	1	1	0	1	1
69	1	1	1	0	1	1	0	1
70	1	1	1	1	1	1	0	1
71	1	0	1	1	1	0	1	1
72	1	0	1	1	0	0	1	1
73	1	1	1	0	1	1	1	1
74	1	0	0	0	0	0	1	0
75	1	1	1	1	0	1	1	1
76	1	0	1	0	0	0	0	1
77	1	1	1	1	1	0	1	1
78	0	0	0	0	1	1	0	1
79	1	0	1	1	0	1	0	1
80	1	0	1	1	0	1	1	1
81	1	0	0	0	1	0	0	0
82	1	1	0	1	1	0	1	1
83	0	1	0	1	1	0	1	1
84	1	1	1	1	0	1	1	1
85	1	0	0	1	1	0	1	1
86	1	0	0	0	0	1	0	1
87	0	1	0	0	0	0	0	1
88	1	0	1	1	1	1	0	1
89	1	1	0	1	0	1	1	0
90	0	1	1	1	1	1	1	1
91	1	0	0	0	0	1	1	1
92	1	0	1	1	0	1	0	1
93	1	0	0	1	1	0	0	1
94	0	0	1	1	1	0	0	1
95	1	0	0	0	0	0	0	0
96	1	1	0	0	1	1	0	1
97	1	0	1	1	1	1	1	1
98	1	0	0	1	0	0	0	1
99	1	1	1	1	0	1	1	1
100	1	1	0	0	0	0	0	1
101	1	1	1	0	0	0	0	1
102	1	1	1	1	1	1	0	1
103	0	1	1	1	1	1	0	0

The following tables show the results of the unequal variance t-tests. We compare all single pair comparisons for each rating.

t-Test: Two-Sample Assuming Unequal Variances	*Performance A-Rated*	*Performance B-Rated*
Mean	0,86579098	0,83088005
Variance	0,11626336	0,14056241
Observations	1751	3193
Degrees of freedom (df)	3900	
t Stat	3,32228496	
P(T<=t) one-tail	0,00045053	
t Critical one-tail	2,3273044	

t-Test: Two-Sample Assuming Unequal Variances	*Performance B-Rated*	*Performance C-Rated*
Mean	0,83088005	0,68640777
Variance	0,14056241	0,21546133
Observations	3193	1030
Degrees of freedom (df)	1486	
t Stat	9,07916664	
P(T<=t) one-tail	1,6925E-19	
t Critical one-tail	2,32885988	

t-Test: Two-Sample Assuming Unequal Variances	*Performance C-Rated*	*Performance D-Rated*
Mean	0,68640777	0,61407767
Variance	0,21546133	0,23727424
Observations	1030	824
Degrees of freedom (df)	1725	
t Stat	3,24399162	
P(T<=t) one-tail	0,00060061	
t Critical one-tail	2,32851153	

Phase 5

H5 - The following tables show the scoring for the descriptions texts of the participants.

Scoring system:
1: complete / very good text quality / 4 or more examples given
2: almost complete / good text quality / 3 examples given
3: partly complete / OK text quality / 2 examples given
4: not very complete / low text quality / 1 example given
5: unprecise / poor text quality / 0 example given
6: missing / very poor text quality / 0 example given

Scores for A rated participants

ID	Distance to canonical class	Functionality explained?	Elements explained?	Roles/functions of elements explained?	Abstracted?	Differentiation to other classes	Subjective scoring for text quality	Score for examples	Average Score
wedel723	0	1	1	1	1	1	1	3	1,285714286
wedel524	0	1	1	1	1	1	1	4	1,428571429
wedel563	0	1	1	1	1	3	2	6	2,142857143
wedel862	0	1	2	2	1	1	2	2	1,571428571
wedel804	0	1	1	1	1	6	1	4	2,142857143
Mais	0	1	1	1	1	2	1	5	1,714285714
Post	0	1	1	1	1	1	1	1	1
Reifen	0	1	1	1	1	6	2	6	2,571428571
Apfel	0	1	1	1	1	1	1	2	1,142857143
Birne	0	1	1	1	2	2	2	2	1,571428571
Eule	0	1	1	1	1	1	1	4	1,428571429
Katze	0	2	1	1	1	1	3	6	2,142857143
Berlin	0	1	3	3	1	3	3	4	2,571428571
Biber	0	1	1	1	1	2	1	2	1,285714286
Bremen	0	1	1	1	1	1	1	1	1
Farbe	0	3	3	3	1	3	3	4	2,857142857
Frosch	0	6	6	6	6	6	5	6	5,857142857
Mean									**1,98**

Scores for B rated participants

ID	Distance to canonical class	Functionality explained?	Elements explained?	Roles/functions of elements explained?	Abstracted?	Differentiation to other classes	Subjective scoring for text quality	Score for examples	Average Score
wedel714	1	3	3	3	2	4	4	6	3,571428571
wedel603	2	3	3	3	1	6	4	4	3,428571429
wedel480	1	1	2	2	1	2	2	4	2
wedel692	1	2	2	2	1	4	3	3	2,428571429
wedel751	1	1	1	1	1	1	1	1	1
Acht	1	2	3	2	1	6	2	1	2,428571429
Affe	1	1	1	1	1	3	2	1	1,428571429
Boot	1	1	1	1	3	3	2	3	2
Telefon	2	1	1	1	1	3	2	1	1,428571429
Blume	1	1	1	1	1	2	2	1	1,285714286
Igel	1	1	1	1	3	3	2	1	1,714285714
Lampe	1	1	1	1	1		1	6	1,833333333
Licht	2	1	2	2	1	3	2	4	2,142857143
Seife	3	1	1	1	1	1	1	6	1,714285714
Wasser	1	1	1	1	1	6	1	3	2
Wolke	1	3	2	3		3	3	1	2,5
Banane	1	1	1	1	1	2	2	4	1,714285714
Feuer	2	1	2	2	1	1	2	6	2,142857143
Kerze	2	2	2	2	1	1	2	6	2,285714286
Koffer	1	2	1	2	1	6	2	6	2,857142857
Hund	3	1	1	1	1	1	1	2	1,142857143

Regen	3	1	1	1	1	3	1	1	1,285714286
Sonne	3	1	1	1	1	6	1	1	1,714285714
Genf	3	3	3	3	3	5	3	2	3,142857143
Golf	1	3	3	3	1	3	3	2	2,571428571
Ampel	1	2	2	2	1	6	6	6	3,571428571
Buch	2	3	5	3	1	6	2	1	3
Bus	1	1	1	1	1	1	1	1	1
Haus	3	1	1	1	1	3	2	1	1,428571429
Kran	3	5	5	5	1	5	5	6	4,571428571
Aula	2	2	2	2	4	6	3	1	2,857142857
Mean									**2,20**

Scores for C rated participants

ID	Distance to canonical class	Functionality explained?	Elements explained?	Roles/functions of elements explained?	Abstracted?	Differentiation to other classes	Subjective scoring for text quality	Score for examples	Average Score
wedel836	4	1	3	2	1	1	2	3	1,857142857
Belgien	5	5	5	5	4	5	4	1	4,142857143
Indien	4	3	3	3	1	6	3	4	3,285714286
Italien	6	1	1	1	1	6	1	3	2
Platz	5	4	4	4	1	3	3	3	3,142857143
Tasche	6	6	6	6	4	6	5	3	5,142857143
Baum	5	1	2	2	1	5	3	3	2,428571429
Hamburg	4	1	2	2	3	6	3	3	2,857142857
Kiel	4	1	1	1	1	1	1	1	1
Tier	5	3	3	1	2	5	3	3	2,857142857
Mean									**2,87**

Scores for D rated participants

ID	Distance to canonical class	Functionality explained?	Elements explained?	Roles/functions of elements explained?	Abstracted?	Differentiation to other classes	Subjective scoring for text quality	Score for examples	Average Score
wedel640	8	3	3	3	2	6	3	3	3,285714286
wedel579	9	2	3	3	4	4	4	4	3,428571429
wedel673	8	3	4	6	6	4	4	2	4,142857143
Bach	8	6	3	6	6	6	4	2	4,714285714
Kuchen	8	3	3	3	4	5	4	6	4
Sand	7	3	3	3	1	6	3	2	3
Torte	9	3	4	4	4	6	3	1	3,571428571
Tulpe	8	4	6	6	6	6	4	2	4,857142857
Eisen	7	5	5	5	2	6	0	6	4,142857143
Elbe	9	5	5	5	6	6	0	4	4,428571429
Elch	7	5	5	5	1	3	0	6	3,571428571
Hase	9	6	6	6	6	6	5	1	5,142857143
Mean									4,02

Appendix B: Sheets for experiment

Klassifikation von interaktiven Grafiken

Phase 1: Paarvergleiche

Entscheiden Sie für je zwei interaktive Grafiken, ob diese der gleichen Klasse angehören.

Alltagsbeispiel für Objekte einer Klasse:

Ford Ka und Renault Twingo sind beides Autos.
Daher gehören sie zur **Klasse der Automobile**.

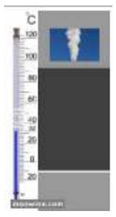
Alltagsbeispiel für Objekte unterschiedlicher Klassen

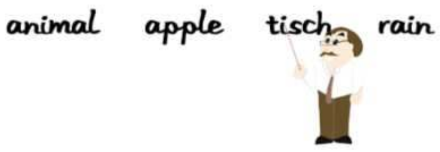
Ford Ka und ein Tisch gehören nicht zur gleichen Klasse.

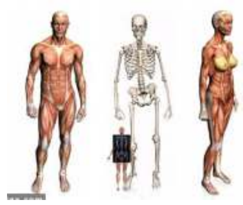
Phase 2: Klassenbildung

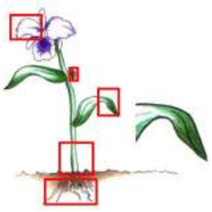
Bitte umkreisen Sie auf diesem Blatt jeweils alle Grafiken, die Ihrer Meinung nach derselben Klasse angehören.

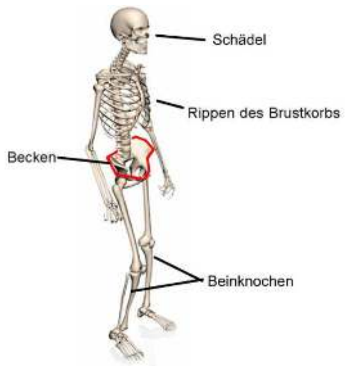
Jede Grafik darf zu mehreren Klassen gehören, d.h. Sie dürfen dieselbe Grafik verschiedenen Klassen zuordnen.





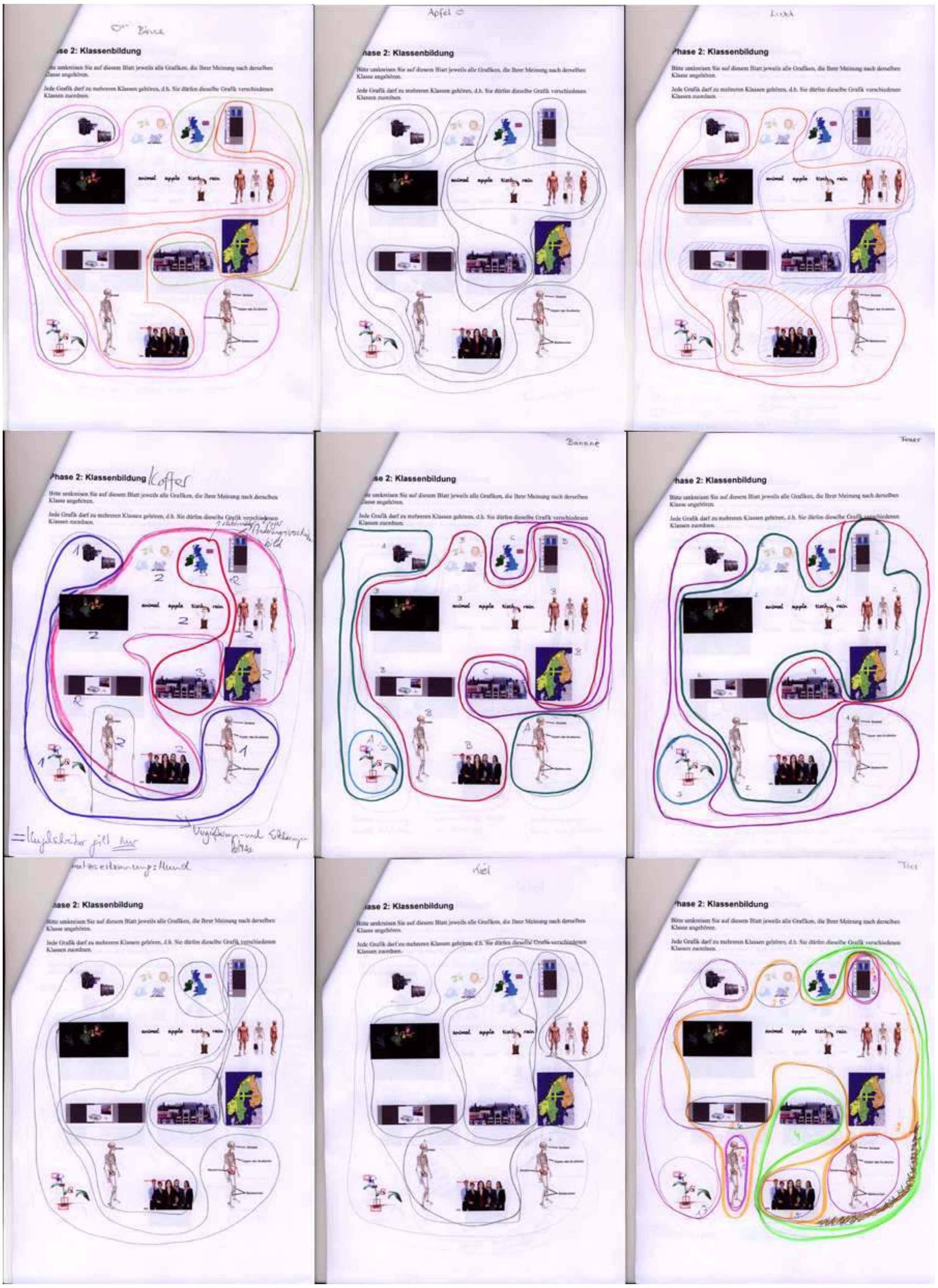






Example sheets from phase 2:

The following example sheets are taken from the first experimental group.

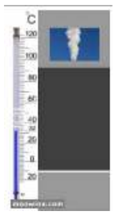


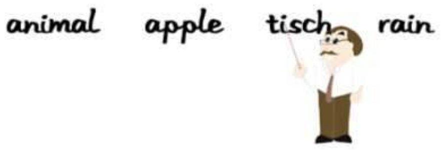
Phase 3: Strenge Klassenbildung

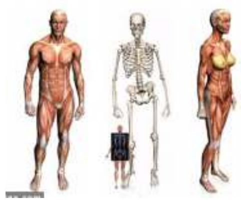
Bitte umkreisen Sie – wie in der vorherigen Aufgaben - auf diesem Blatt jeweils alle Grafiken, die Ihrer Meinung nach derselben Klasse angehören.

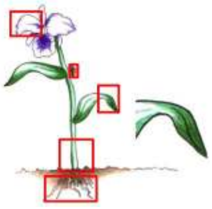
Dieses Mal ordnen Sie bitte jede Grafik eindeutig einer Klasse zu.
Jede Grafik darf nur zu einer Klasse gehören!

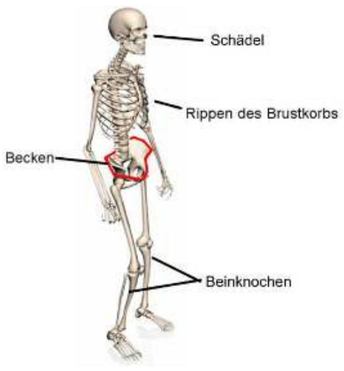
Wenn Sie sich bei einer Grafik nicht sicher sind, so ordnen Sie diese der Klasse zu, in die sie am Besten hinein passt.





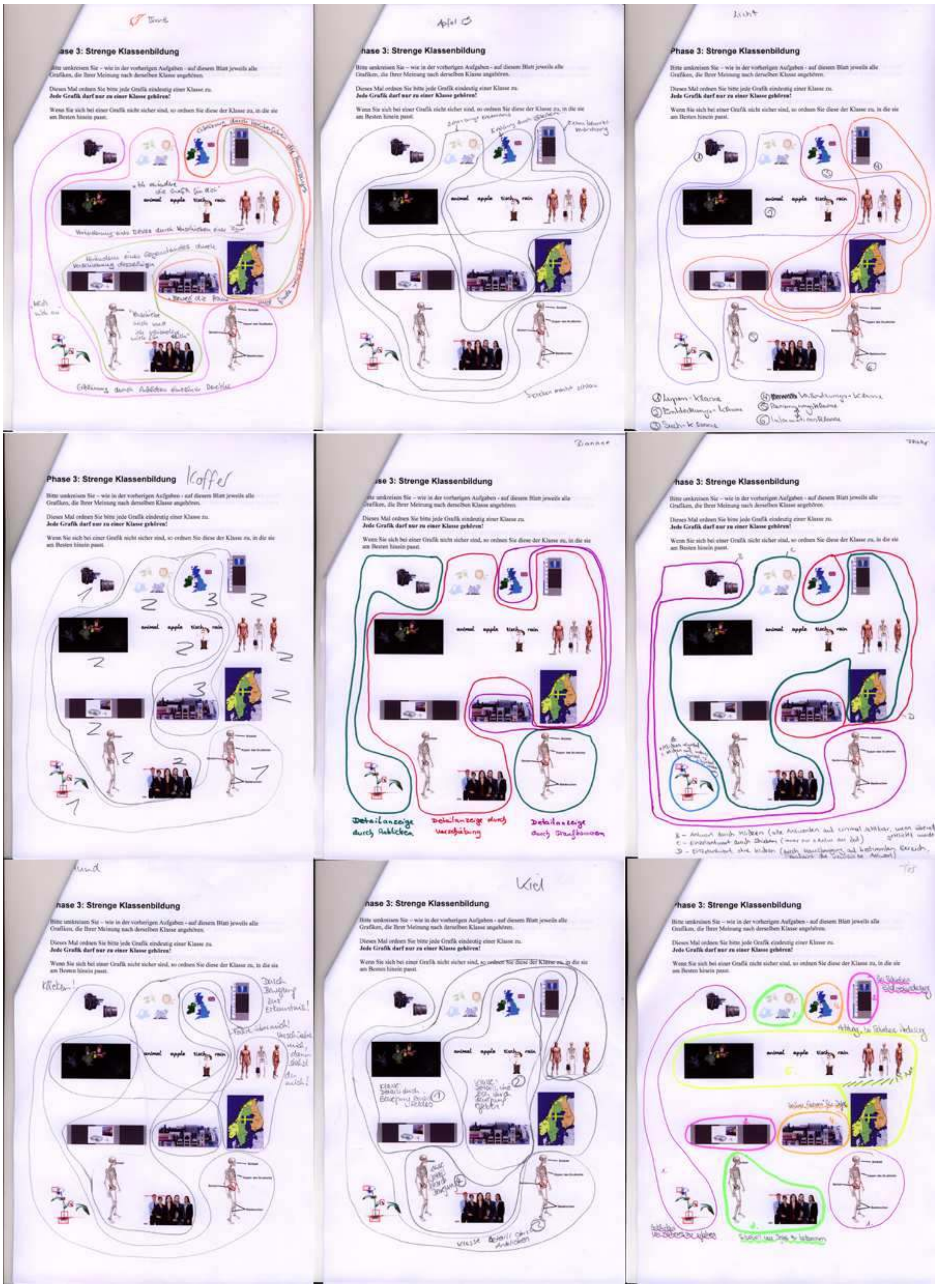






Example sheets from phase 3:

The following example sheets are taken from the first experimental group.



Phase 3: Strenge Klassenbildung

Bitte umkreisen Sie – wie in der vorherigen Aufgaben - auf diesem Blatt jeweils alle Grafiken, die Ihrer Meinung nach derselben Klasse angehören.

Diesen Mal ordnen Sie bitte jede Grafik eindeutig einer Klasse zu.
Jede Grafik darf nur zu einer Klasse gehören!

Wenn Sie sich bei einer Grafik nicht sicher sind, so ordnen Sie diese der Klasse zu, in die sie am Besten hinein passt.

animal
apple
tisch
rain
Koffer
Kiel
Detailanzeige durch Anblicken
Detailanzeige durch Verschiebung
Detailanzeige durch Draufklicken

Phase 4:

Verwenden Sie den letzten Aufgaben-Bogen (Phase 3) und vergeben Sie für jede Klasse einen Namen. Der Name sollte typisch für die Interaktion sein und darf gerne eine Alltags-Analogie haben.

Schreiben Sie den Namen bitte direkt an die eingezeichnete Umrandung der Klasse.

Phase 5: Beschreibung einer Klasse

Entscheiden Sie sich für die Klasse, von der Sie am meisten überzeugt sind und beschreiben Sie stichpunktartig:

- Wie funktioniert diese Interaktion?
- Was für Elemente sind daran beteiligt?
- Welche Funktion haben die Elemente?
- Was unterscheidet die Interaktion von den anderen Klassen?
- Wofür könnten Sie diese Interaktion noch einsetzen? Nennen Sie Beispiele!

Name der Klasse:

Beschreibung der Klasse:

Appendix C: The generalized birthday problem

To assess the significance of the experimental classification results we need to find out how likely it is that the same classes have been identified by several participants by pure chance. If there are n possible classes (e.g. 16384) and the participants have identified m classes (e.g. 377) we can ask how likely it is that exactly the same class has been identified by more than one participant.

Since we want to know whether any of the m identified classes matches any of the other ones, we have to compare the first class with all of the other classes, the second class with all of the remaining classes and so on. This is similar to the birthday problem[35] that asks how likely it is that in a group of m people 2 persons have the same date. The number of possible dates (or classes) would be in this case n=365.

We cannot simply use a standard formula of the birthday problem because we are not interested in the probability of identifying the same class 2 times but up to 43 times. The formula becomes already very complicated for the case of only 3 identical classes (birthdays). Since it is not our objective to develop a complicated statistical formula, we choose to simulate the problem and see how many times the event (the same class has been identified by x participants) could occur by chance.

To do this we simulate 100 000 virtual experiments. In each virtual experiment, we will randomly draw m (numberOfChances) out of n (numberOfPossibleClasses) possible classes. For example, in each run of the simulation we draw 377 classes out 16384 classes. Since each participant can identify any of the possible classes (in theory a participant could even identify the same class multiple times), each draw can be any of the 16384 classes. Then we are counting for each of the possible 16384 classes how many times it has been drawn by chance. In most cases the class will not be drawn at all. If the identification of classes is truly random (as in the simulation) nearly 377 classes will be drawn only one time each. However, in some cases the same class might be drawn more than one time by chance.

The probability that the same class is drawn three times by chance is less than 4:100 (or p=<0,04). The probability that the same class is drawn four times by chance is less than 3:10000 (or p<0,0003). The probability that a class is drawn more than four times by chance is much lower accordingly; in fact we run 100 000 simulations and it occurred not once. The low probability for such chance occurrences is due to the high number of possible classes.

For the experimental phase in which the complete partitions have been compared, the probabilities are even lower because there are 190 899 322 different complete partitions. We did run the simulations 1 000 000 times and the same partitions (or index numbers) have never been drawn more than three times by chance. This shows that identical classifications are not caused accidently.

The commented source code for the simulation can be found on the next page (written in Java).

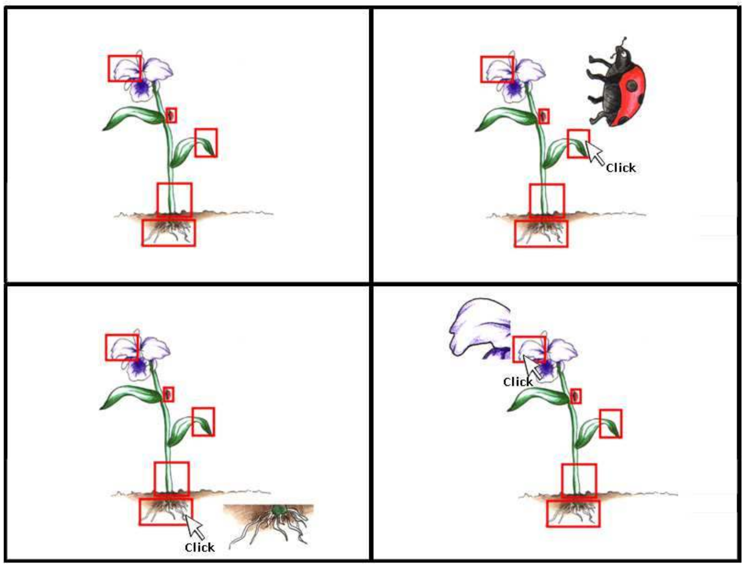
[35] More details about the birthday problem can be found, for example, in the Wikipedia: http://en.wikipedia.org/wiki/Birthday_problem

```java
/** Simulates a generalized situation of the birthday problem */

public class BirthdayProblemSimulation {

   public static void main (String args[] ) {

       final int numberOfPossibleClasses = 365; // Number of possible classes (or possible days: 365)
       final int numberOfChances = 100; // Number of potential hits (or people in the room)
       final int numberOfSimulation = 100000; // number of simultaion, each simulation runs through all chances

       // numberOfPostiveCases counts how many times the following events occur:
       // how many times has a class been found 2 times (2 persons in the room have the same birthday)
       // how many times has a class been found 3 times (3 persons in the room have the same birthday)
       // ... etc.
       // the highest possible number is the numberOfChanges. if n classes have been identified the max number
       // of identical classes is n, i.e. if 20 people are in the room the max number of people who share a birthday is 20
       int [] numberOfPostiveCases = new int [numberOfChances];

       System.out.println("Starting simulation");

       int randomNumber;

       // run through the specified number of simulations
       for (int simulationCounter = 0; simulationCounter < numberOfSimulation ; ++simulationCounter) {

           // distribution of classes. Each potential class can be found 0..numberOfChances times
           // (a year has 365 days; on each day 0..number of people in the room could have birthday)
           int [] verteilung = new int [numberOfPossibleClasses + 1];

           // run through number of chances / identified classes (check for every person the the birthday)
           for (int chanceCounter = 0; chanceCounter < numberOfChances; ++chanceCounter){

               // random assignement which actual class has been identified by a participant
               // (random assignement of one of 365 days as birthday for one person)
               randomNumber = (int ) ( Math.random() * numberOfPossibleClasses );

               // class "randomNumber" has been identified one more time
               // (the x. day of the year is the birthday of person y. hence, one more person has birthday on the x. day)
               verteilung [randomNumber] ++;

           }

           // We want to know whether any class (birthday) has been identified
           // 0,1,2,3,4...numberOfChances times as the same
           // (0 person have birthday at the same date, 1 person has birthday at the same date
           //  2 persons have birthday at the same date, ..., max. all people have birthday at the same date )
           boolean [] positiveEreignisseAktuellerDurchgang = new boolean [numberOfChances];

           // check for all potential classes (all potential birthdays of the year)
           for (int check=0; check<numberOfPossibleClasses ; check++ ) {

               positiveEreignisseAktuellerDurchgang [ verteilung [check]  ] = true;

               // verteilung [check] has stored the number of times a potential class
               // has actually been identified. In most cases that would be 0 times (in a small group for most
               // days of the year no one has birtday on that date)
               // For example, verteilung [42] tells us how many people have identified the class with the index 42
               // (or how many people have birthday on the 42nd day of the year)
               // we are interested wether there are any classes (days of the year)
               // that have been identified 2,3,4... or more times. So, if one or more classes have
               // been identified 2 times positiveEreignisseAktuellerDurchgang[2] will be set true
               // (the event that two people have their birthday one the same date is set to true)
               // likewise, if one class has been identified 10 times (10 people have the same birthday), then
               // positiveEreignisseAktuellerDurchgang [10] is set true
           }

           // increase the number of positive events that have been found
           // for example if any class has been found 2 times (two persons have the same birthday)
           // then the number of positiveCases [2] is increased as there is one more
           // case in which we have two identical classes (or 2 persons sharing one birthday)
           // Likewise if any class has been identified 3 times in this simulation run, then
           // postiveCases [3] is set to true, etc.
           for (int n = 0; n < numberOfChances ; ++n) {
               if (positiveEreignisseAktuellerDurchgang[n]) {
                   numberOfPostiveCases[n]++;

                   // the positive event n occured, i.e. at least one class has been
                   // identified n times
                   // therefore, we increase the n.th positive case
               }
           }
       }  // end of one simulation run

       // Output of results. Provide for each event the frequency/probability of cases in which
       // the same class has been identified 0,1,2,3...numberOfChances has been identified in n cases)
       // (how many times did the evet that 0,1,2,3...number of persons in the room have the same birthday)
       for (int i = 0; i < numberOfChances ; ++i) {
           // only report the events that actually occured
           if (numberOfPostiveCases [i] > 0) {
               // print the number of actual events in relation to all simulation runs
               // for example, if in 100 000 simulations, we had 50 runs in which the same
               // class has been identified 6 times, then
               // the frequency of having 6 identical classes is 50 / 100 000 = 0,0005
               // that means the probability that the same class has been identified 6 by chance is 0,05%
               int frequency = numberOfPostiveCases[i];
               double relativeFrequency = ((double) frequency) / ((double) numberOfSimulation  );
               String ausgabe = "Ereignis: "+i+ " gleiche: " + frequency + " ("+relativeFrequency+"%)";
               System.out.println(ausgabe);

           }
       }
   }
}
```

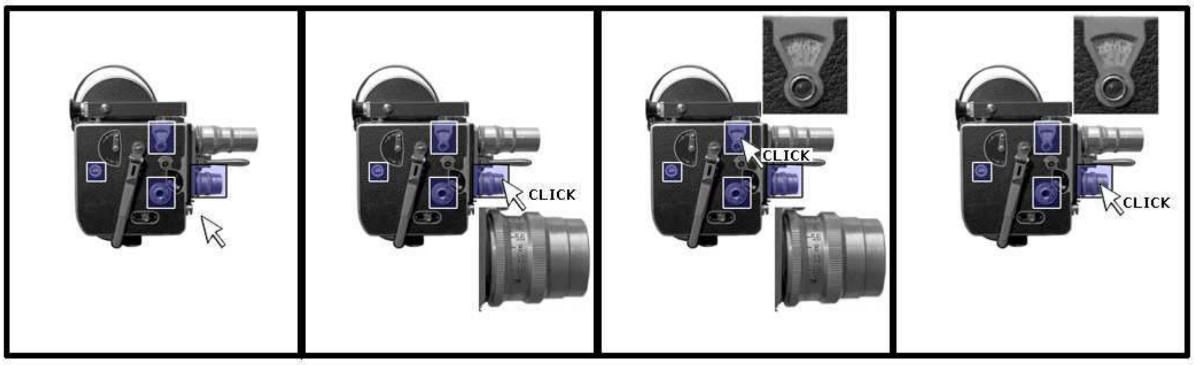
Appendix D: Used graphics

Pattern P1

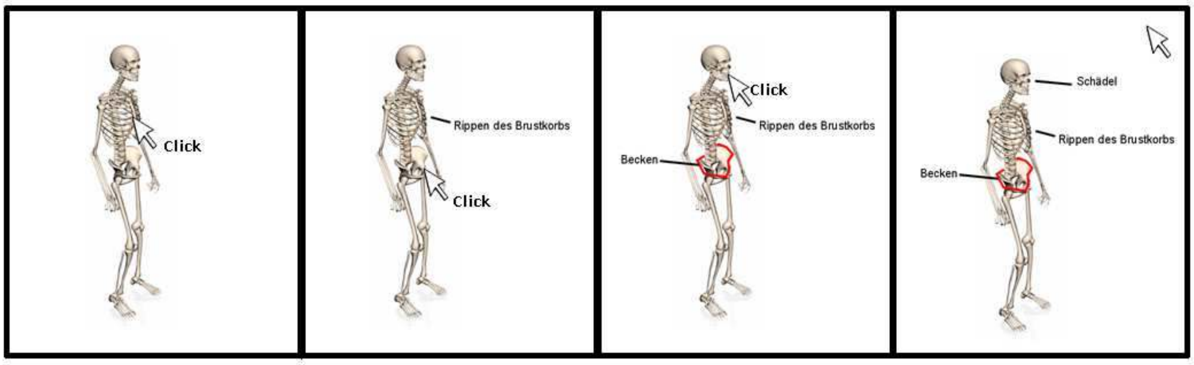
A_{P1}: If the user clicks at on of the marked areas, an enlarged illustration pops out. Any enlarged illustration shown before disappears. Thus, only one zoom is visible at a time and the attention is directed to only one zoomed detail.



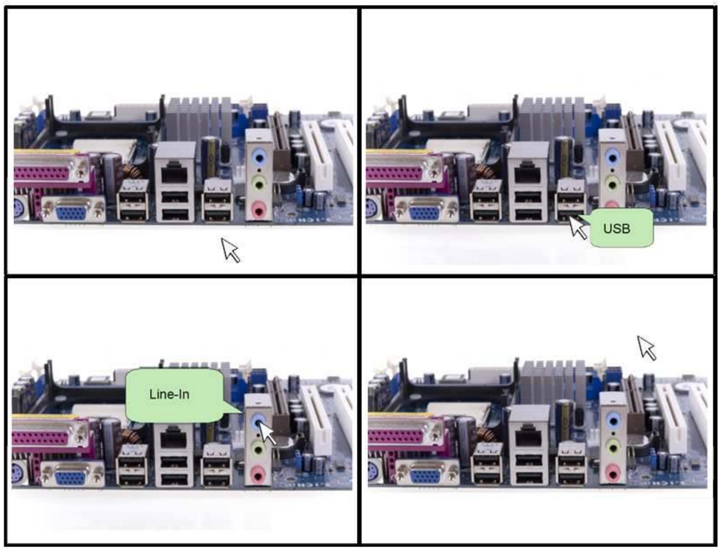
B_{P1}: If the user clicks at on of the marked areas, an enlarged illustration pops out. The detailed illustration has to be deactivated explicitly by clicking at the corresponding area for another time.



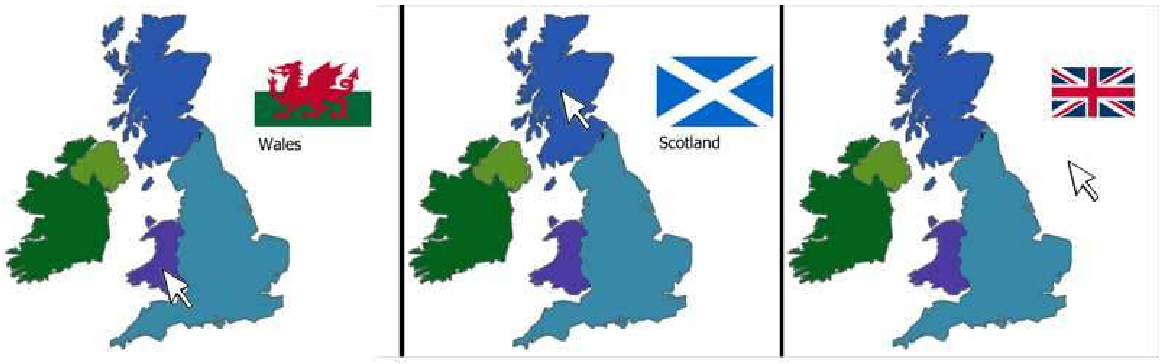
C_{P1}: By clicking at areas of the skeleton, the labels appear.



D$_{P1}$: If the mouse pointer moves over interesting areas of the image, a text bubble appears. The text in the bubble explains what is shown under the mouse pointer. The text bubbles disappear automatically as the mouse moves on to other interesting areas.

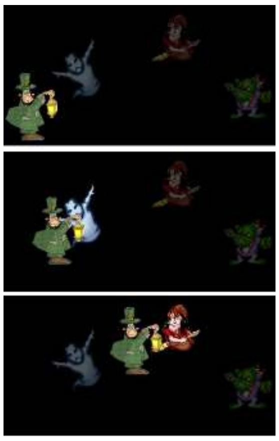


E$_{P1}$: If the mouse pointer moves over one of the four states, the flag of the state is show. If the mouse pointer is over none of the states, the flag of Great Britain is shown.

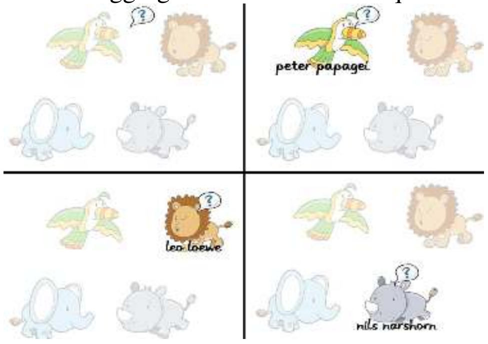


Pattern P2

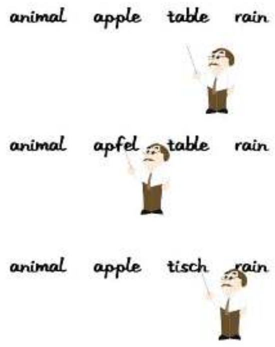
F$_{P2}$: Dragging the night watchman over the images illuminates them.



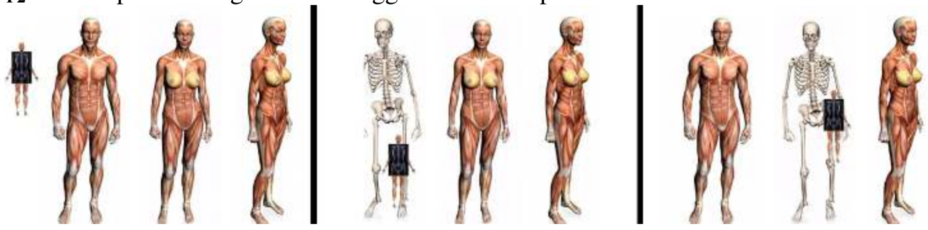
GP2: Dragging the bubble with the question mark over an animal shows its name.



H$_{P2}$: Pointing at a vocabulary translates it.

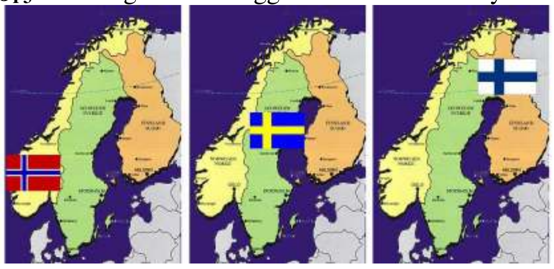


I$_{P2}$: An inspector image can be dragged over each person and the skeleton is shown.

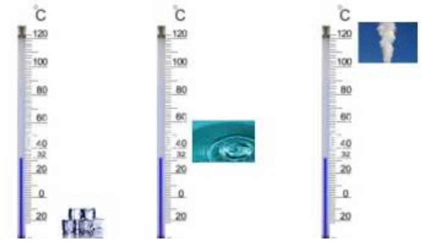


Pattern P3

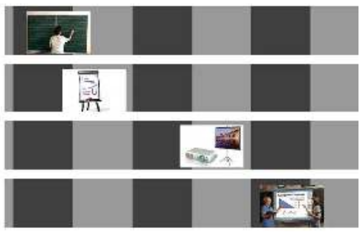
J$_{P3}$: The flag can be dragged over each country. It shows the flag of the country dynamically.



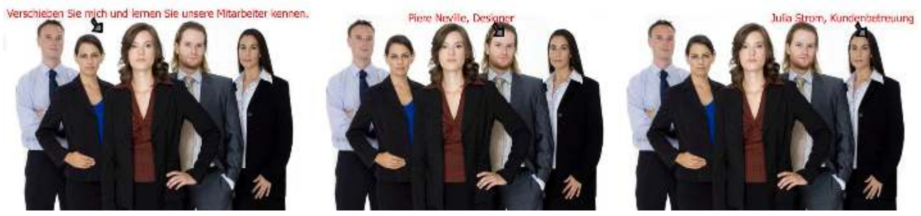
K$_{P3}$: the state of water changes according to the temperature.



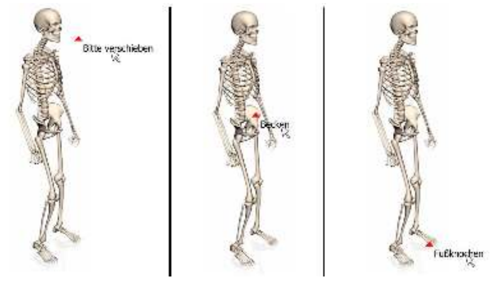
L$_{P3}$: The picture can be dragged along a timeline and shows several development states of blackboards and whiteboards.



M$_{P3}$: The text can be dragged over each employee and shows his/her name and job description.


Verschieben Sie mich und lernen Sie unsere Mitarbeiter kennen.
Piere Neville, Designer
Julia Strom, Kundenbetreuung

N$_{P3}$
By dragging the label over different bones, the name of the bone is shown.



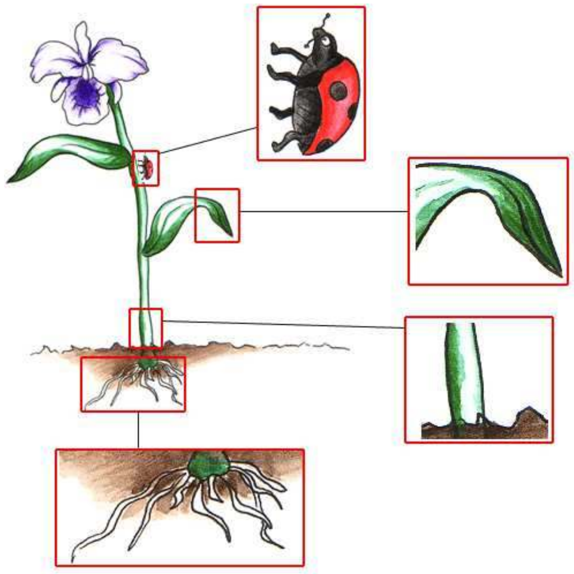
Appendix E – Patterns of interactive graphics

The Switch problem (Master thesis)

1. Synopsis

Illustrations can be labelled with text. To set the focus on one text label all other labels should be invisible or use a low opacity level. Furthermore some parts of the illustration can be displayed enlarge by clicking at small thumbnails.

2. Figure



The Switch problem

3. Description

The example shows an illustration of a flower which is a simple image element. For some areas on the flower illustration descriptive text labels are available. By default all text labels are invisible. If the user clicks on an area the label that belongs to it becomes visible. That is the text label was switched ON. If the user clicks on another hot area the related text becomes visible. Optionally all other text elements can be set back to their invisible states. The user could either successively turn on all text labels or let only one label be visible.

Besides the descriptive text labels there are some details of the illustration that could be shown enlarged. At the beginning all enlarged clips are invisible. If you click at the end of the right leaf the image element showing the enlarged leaf becomes visible. Clicking at the image or at the illustration clip will hide the image again. Hence the illustration clip works as a switch that changes the state of another image element depending on its current state.

4. Action Trigger and Response

The visual language should offer a way to hold two different states for an element. Technically this could be a pair of two snapshots. A switcher should offer to activate a certain state or to switch to the opposite state. Action commands can trigger the switch, e.g. sending messages to the switch:

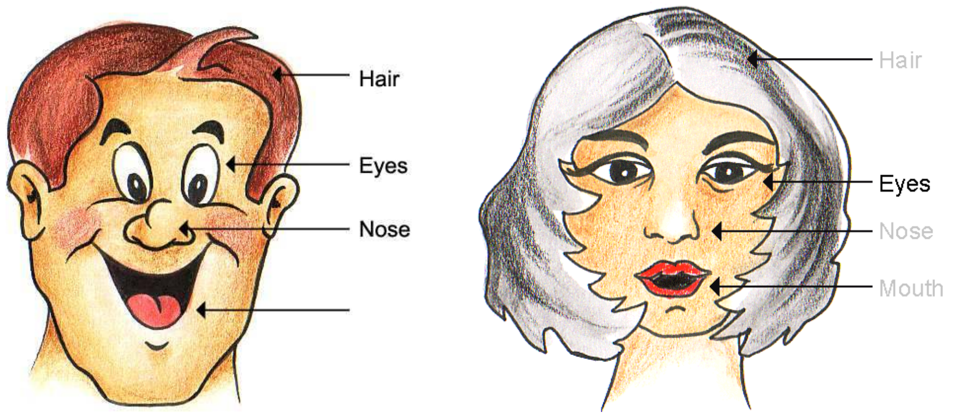
- Turn ON
- Turn OFF
- Turn to opposite state

In addition a way should be provided to link switchers. If one switch was turned on all other switch can turn off automatically. A chain of switchers could also offer services such as turning all switches ON or OFF. If the chain also defines an order for the switches then one switch after another could be activated.

Since the property changes for all switches in a chain are most likely to be equal it is helpful to use the same screenshot for all of them. For example to show or hide an element always the visible property is set. To set a focus on a text element the colour of the element could be set to red. On loosing the focus the colour is reset to black. If a set of switches for text elements is chained then all elements could use the same shot that colours them red or black. For the element that is turned ON (receive focus) the shot holding the red colour is activated, all other elements are turned OFF. So the shot holding the black colour is used for each of them.

5. Related problems

Giving optional information on a slide is a good way to keep slides clean. Also focuses can be set (B). Context-sensitive information can be given. In figure (C) a head is shown. The text labels are turned ON one after another; the labels 1, 2, 3 have already been activated. Using switches different level of details can be given on a slide. Switches can also be used to provide checkboxes on a slide.



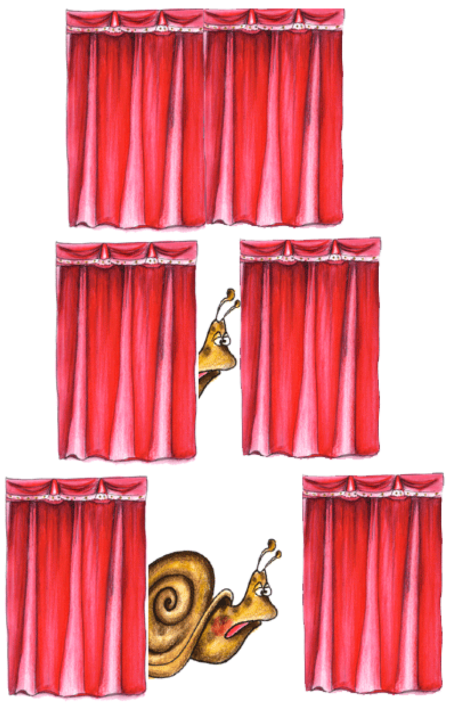
Labels can be made visible step by step (man) or a focus can be set (woman).

Movement relations / The Curtain problem (Master thesis)

1. Synopsis

Based on the problem of having two curtain halves moved into opposite directions we can handle a class of problems that involve element related movements.

2. Figure



3. Description

The curtain problem uses two image elements each showing one half of a curtain. Dragging one of the two halves will move the other half automatically into the opposite direction. In the beginning of this chapter we discussed which Tcl code was necessary to implement such behaviours.

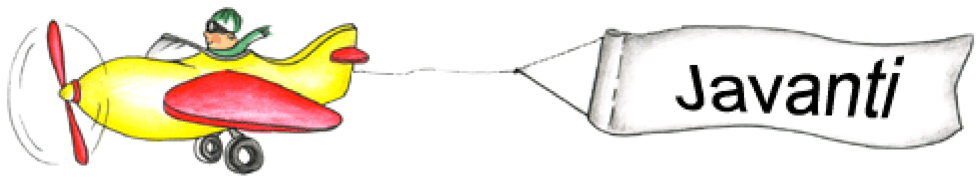
Whenever one element's movement is related to other moving elements an imperative program would solve the problem by performing those operations: read the location values of the leading element, compute new values for the following element and finally assign the calculated values to the element.

Movement relations between elements are an easy to understand yet powerful way to add interactivity

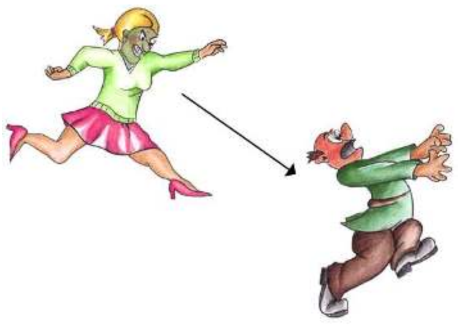
4. Action Trigger and Response

To implement a movement relation the user can add a follow-move behaviour to the element that should be guided. One of the behaviour's parameters is the leading element. Other parameters specify the follow type, e.g. directly, delayed, opposite, rotated, or an acceleration factor. In the next sections some related examples explain more parameter settings. The leading element can be set to dragable so that the user can manually move the element later on the slide. In the curtain example it make sense to apply the same behaviour to both elements because it does not matter at which side you open the curtain. In other examples the leading movement is restricted to one element.

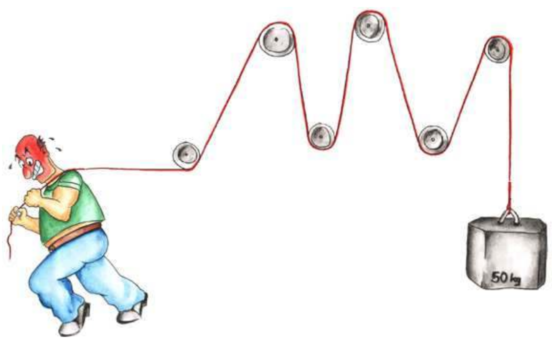
5. Related problems



The simplest case for a related movement is a 1:1 follow-up of a second element. Example (a) shows an air plane that can be moved on a slide. A banner follows the slide. If you add the follow-up behaviour and a behaviour that move the banner element randomly up and down it looks like the banner flaps in the wind.

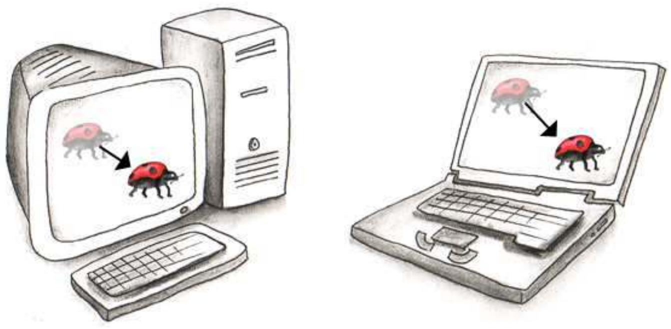


Example (b) shows a man that is hunted by his wife. A follow-up behaviour is added to the wife, which lets her move along the shortest direct path. By setting an acceleration we can make sure that the woman will finally catch her husband. Vice versa a negative acceleration can let the man escape enlarging the distance every single step he takes. Note that the woman element should only move if the man element is moving, too.

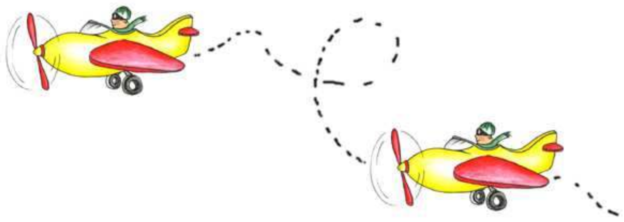


Example (c) shows a man (one image element) using a lifting block to pull up a heavy stone (another image). Moving the men to the left will move the stone upwards. The follow-up movement is rotated about 90 degrees. If we use the behaviour in reverse for the men then the stone could also move down and pull the men to the right. Adding an additional move-down behaviour to the stone would automatically move the stone downwards unless the user is dragging the men (assuming that dragging elements has a higher priority then the move-down behaviour).

Example (d) shows a race between a snail and a turtle. Both animals start at the same point. The follow-up is a direct one with acceleration. That is moving the turtle will also move the snail slower. Moving the snail would move the turtle accelerated.



Example (e) is a slide that demonstrates desktop sharing. Moving the rectangle on one of the two screens will move the other element in exactly the same way. However a delay for the movement will apply.



Example (f) shows a follow-up along a path. After a short time delay the second plane will move along the same positions as the first plane did.

Dynamic Labelling (PLoP 2006)

Name: Dynamic Labelling

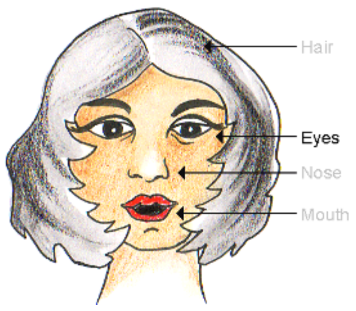
Problem overview:
Physical space limits the number of possible labels. Too many labels increase the complexity and make it harder to interpret the graphic.

Use when:
- A lot of components of the graphic need to be labelled.
- Static labelling makes the graphic too complex or unreadable.
- Draw attention to only one or a limited number of components at a time.
- Provide explanatory verbal information in local contexts.
- Reduce the complexity of a graphic.
- You want to suggest an optimal sequence in which to read the graphic.

Forces:
- The user has to understand how to activate a dynamic label.
- The user may want to have more than one component to be labelled at the same time.
- To see all labels the user has to activate each label manually.
- Labels can hide parts of the relevant information in the graphic.
- Readability of labels depends on it visual surrounding.

Solution summary: Dynamic labels appear on the screen only when needed.

Diagram:



Solution details:
The form of a dynamic label could be plain text that appears above the image area that is subject to be named or explained. You can also use a line or an arrow that links the image area of interest to a popup text that is placed somewhere else. By doing so you avoid that the relevant area is covered by the text and the graphic does not interfere with the text. If text is placed besides (and not over) the relevant area you can also put a box behind the text to improve readability. A good combination of a textbox and a pointing arrow is a text bubble.

Of course the dynamic label has not to be text. You can also show dynamically sub graphics within the main graphic. For example the popup graphic could show a different view, further visual explanation or DETAILS ON DEMAND of the labelled area. Labels can show an enlarged picture of a selected area, and are one approach for pre-defined ZOOMING.

Activating the labels should be made easy for the user. First of all the user has to know that dynamic labels are available. Second, he needs instructions where to find and activate them. If you have a single graphic using dynamic labels, you should write a hint that dynamic labelling is available. If you have many graphics with dynamic labelling, e.g. a web based training course, you should give a short instruction at the beginning. The instruction could include animated DEMONSTRATION.

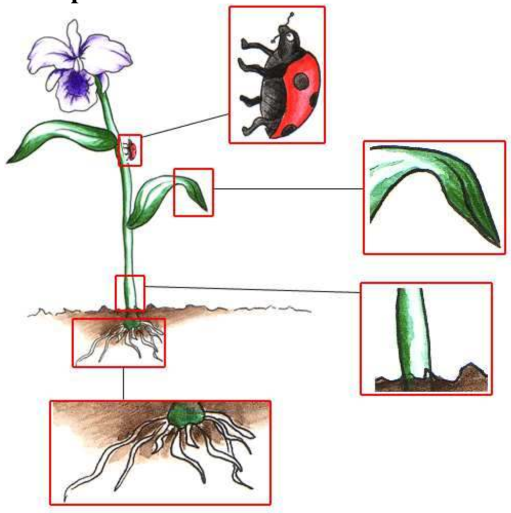
The simplest way to activate a dynamic label is a ROLL-OVER with the mouse. The user finds out very quickly where labels are by moving the mouse over areas he is interested in. He may not find all labels, but he finds all labels he is interested in. The mouse pointer acts - as the name says – as a pointer. Thus, connection lines between the relevant area and the label are not required. The label could appear even outside the graphic. As soon as the mouse exits, the dynamic label disappears, which is useful to immediately reduce complexity again. However, if the label needs to be shown for a longer time frame, keeping the mouse pointer within the relevant area can be difficult. It also prevents the user from doing other tasks without loosing the label. Showing more than one label is not possible using mouse roll-overs.

To keep a label permanently visible you can use a SWITCH BETWEEN OBJECT STATES. If the user clicks at the relevant area, the label appears. To let it disappear the user has to switch it explicitly off. This could be done by clicking at the label or at the relevant area. This way more than one label can be displayed at the same time. Another option is to switch the label off as soon as another label is selected. This allows only one label at a time. Showing only one label can be useful in presentations to direct the audience's attention and HIGHLIGHT INFORMATION. If there are many dynamic labels or each label is large in size then multiple viewed labels would overlap. Automatically switching off one label as soon as another is switched on reduces the mouse clicks to move on from one label to another.

If activation of dynamic labels is done by clicking at HOT AREAS, the user needs to know where these clickable areas are. You can provide visual clues such as drawn bounds or small circles to indicate where label activation is possible. Many graphics, however, implicitly show areas, e.g. in a map the bounds for different labelling are quite obvious. You can also use ROLL-OVER and a SWITCH BETWEEN OBJECT STATES in combination. Moving the mouse over an area can popup the label temporarily and clicking the mouse will keep it permanently. The roll-over label could actually differ from the "permanent" label: it can act as a label preview (for large sized labels) or as an indicator that a dynamic label is available (e.g. a "click here" popup or highlighting the bounds of the HOT AREA).

Implementation:
Most authoring environments allow you to dynamically show or hide components and you can implement the solution using a SWITCH BETWEEN OBJECT STATES. On the contrary, presentation software usually does not provide features to switch on and off parts of a graphic. However, most presentation tools allow navigation and hyperlinking. If you run through a predefined sequence of dynamic labelling, you can just multiply the same graphic on a sequence of slides and use only one label per slide. The standard functions to step forward/back will guide the user from one label to the next. If each label should be accessible at any time you must provide hyperlinks on all slides showing the graphic. For the hyperlinks, use invisible objects (with transparent opacity) and place them at the areas where a mouse click should pop up the label. This method, of course, is only practicable if the number of labels is small.

Examples:



Dynamic labelling is used to show details of the flower. If the mouse enters one on the small red boxes on the flower, then one of the larger boxes showing the details pops up. The enlarged views are linked to the area they show by a line.

Rationale:
There are psychological reasons to use dynamic labelling. The working memory is limited and too much irrelevant information can cause cognitive overload. Dynamic labelling can provide all needed information just when needed. Also you can place each label closer to the related area of the graphic, and thereby you follow the contiguity principle [CM03].

Selectivity of attention is another reason to use dynamic labelling. Using only some instead of many labels gives hints to the learner to which areas he should pay attention. This is especially useful in presentations. The appearance of a new object in the visual field attracts attention itself [HY94].

Related patterns:
This pattern can be used by: DETAILS ON DEMAND, HIGHLIGHT INFORMATION, OBJECT REPRESENTATION, ZOOMING
This pattern can use: DEMONSTRATION, HOT AREAS, ROLL-OVER, SWITCH BETWEEN OBJECT STATES

Synchronize Object Motion (EuroPLoP 2007)

Intent:

Show exactly how the motions of objects depend on each other.

Problem:

How can I demonstrate the correlation of objects in motion? Cause and effect are difficult to interpret in static illustration because the dimension of time is missing. Correlations in motion cannot be perceived if snapshots only are provided because information about motion paths and speeds between two snapshots is not provided. Arrows can indicate similarity in paths but do not cover variations in speed or time delays. There is no commonly understood semantic for static arrows to map exact information about variation in speed or paused motion. In animations, the user cannot control the motion and has no chance to experiment with differing parameters. Producing such animations is time-consuming if you have to define the motion for each object individually in spite of their dependencies.

Consider this:

- Often, there are implicit motion relations between objects but these only apply if one of the objects is explicitly set into motion. So what triggers the motion of objects? How can you reveal motion relations between objects?

- Very often multiple objects are in motion at the same time, but with today's user interfaces you can usually only control one object directly with a (mouse) pointer. So, how can you control many objects simultaneously?

- In many cases, multiple objects have to perform identical or similar moves at the same time or with time delays. You could set an animation path for each object individually but this is time consuming and a possible source of error. You may define motion paths according to rules that you have in mind. But if you could express the rules to the machine rather than the results only, ad-hoc creation of new animations would be possible and the definition process becomes more efficient and flexible.

- By providing rules, objects can follow other objects, but how can you track the exact path of a moving object? How do you handle variations in speed or direction?

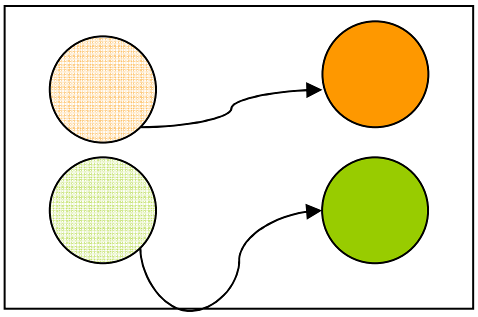
- Objects can move into different directions but using the same motion paths. The animation vectors may look totally different but still base on similar paths.

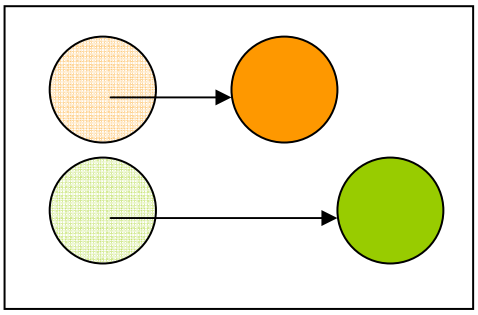
- Varying object scales must be respected. Sometimes objects are supposed to move exactly the same paths but they move in differently scaled coordinate systems. Hence, to map the same distance in two different coordinate systems the actual motion on the screen must differ in the step size measured in pixels.

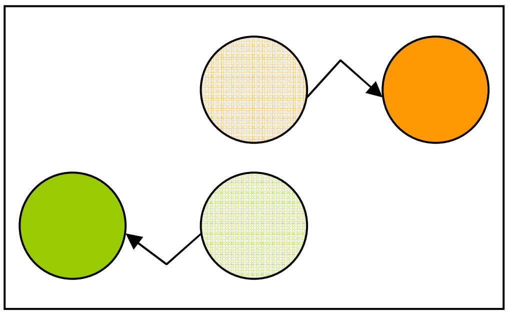
Therefore:

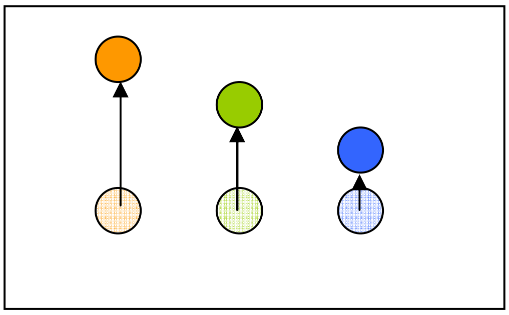
Relate *Guide* and *Fellow* objects that synchronize their motion automatically. Each move of a *Guide* object is captured as a vector and applied to all *Fellow* objects. The motion vector can be identical or transformed, i.e. scaled, rotated or delayed.

Diagram:


The fellow object (green) moves synchronized with the guiding object (orange).

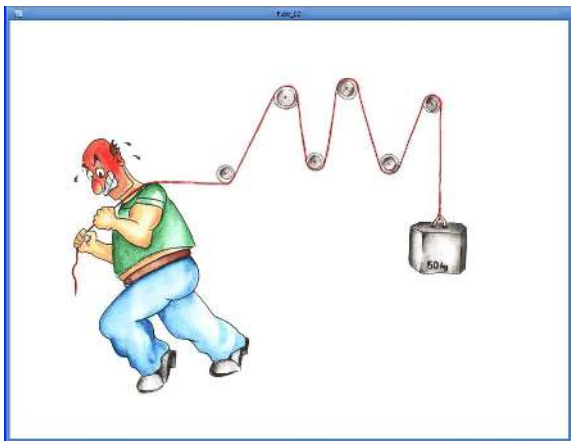

The fellow object moves faster than the guiding object, but follows the same path. A motion scale factor applies.

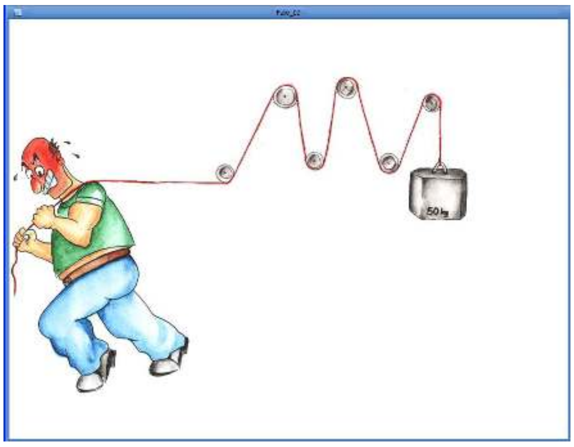

The fellow object performs the same motion as the guiding object, but in a different direction.

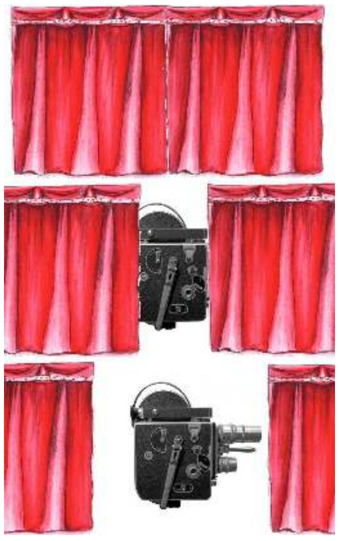

The fellow objects perform the same motion as the guiding object. However, each fellow object starts its motion with a delay.

Examples:

Demonstration of a lifting block. The user pulls a person horizontally and the block moves vertically in synchronisation.

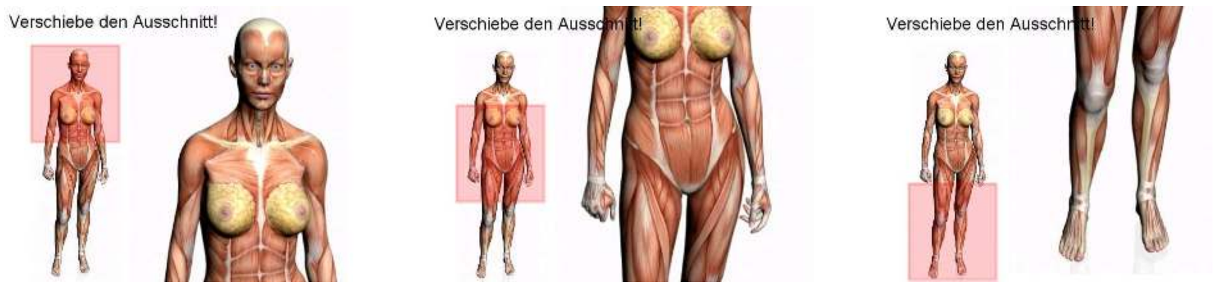






Hiding objects. A curtain is opened and reveals some hidden content. You can drag at each side of the curtain horizontally.

Zoom-Navigator. The red rectangle selects the part of the zoom image that is displayed. To do so, the zoomed image moves into the opposite direction when the rectangle is dragged. Because it is three times larger, it moves three times faster.


Verschiebe den Ausschnitt!
Verschiebe den Ausschnitt!
Verschiebe den Ausschnitt!

When can I use this?

- Objects should move exactly in the same way.
- Objects move in a similar way.
- Object motions relate to each other.
- Show cause and effect of object motions.
- Show that one object drags another object.
- Use a small object to navigate a large object.
- You need sliders or scroll bars to position objects.
- To compare relative speed between objects.
- Move objects automatically in opposite directions
- Scale, rotate, or delay the motion of two objects.

Rationale

Objects that move in synchronisation are perceived as one unit according to the law of common fate [Gol02] defined by gestalt psychologists. If you consider motion to be just an attribute of a visual object, you can also argue that SYNCHRONIZE OBJECT MOTIONS groups objects by similarity of an attribute (law of similarity). If the motion is transformed (scaled, rotated or delayed), then synchronisation can also support the perception of causality. People are very sensible in recognizing relationships between moving objects, in particular they can identify which object causes the motion of another object [War04a]. Synchronizing objects also invites students to experiment with object motion and helps to discover relations between objects, thus allowing “learning-by-doing” [Rie90b]. The synchronisation of small and large object representation is used in navigation panels and scroll bars [Mye90]

Roles, States and Interaction

There are two different object types, the *Guide* object and *Fellow* objects. The related objects can transfer the *Guide* role temporarily among each other.

Guide Object:
There can only be one *Guide* object at a time. The motion of the *Guide* object is exactly captured in a motion vector. The motion vector is then applied to all related *Fellow* objects either immediately or delayed. Also, the motion can be transformed, i.e. scaled or rotated.
You can either define that only one of the related objects can inhabit the *Guide* role. In this case only the motion of this object is captured and applied to the other objects. Another option is to transfer the *Guide* role between all participating objects on demand. In this case the *Guide* role is inhabited by the object that is currently moved by an outside force, i.e. the motion is caused by dragging an object with your mouse pointer or another pattern like *Transport Objects*.

Fellow Objects:
Whenever the *Guide* object moves, the *Fellow* objects move, too. In a group of objects, there is always only one guiding object, but one or more *Fellow* objects. All *Fellow* objects move when the guiding object moves. Motion of *Fellow* objects does not affect the guiding object. However, an object can switch between the roles *Guide* and *Fellow*. Since an object can never be *Fellow* and *Guide* at the same time, *Fellow* motions will never guide any other objects. Thus, cyclic synchronisation is avoided.

It is possible for an object to be a *Fellow* of group A and the *Guide* of group B when A and B are disjunctive, that is no member of group A is a member of group B.

Scaling motion:
To respect different scales and speeds of objects, the motion vector of the *Guide* and its *Fellows* can be scaled by a scale factor. The scale factor defines the relative step size between object A's and B's motion. For example, let the scale factor be 2 and object A moves 15 pixels. Then object B moves automatically 30 pixels. On the other hand, if B moves 50 pixels, then A moves 25 pixels.
Varying scale factors can be used for different object relations if you define multiple *Guide-Fellow* relations. The semantically meaning of the scale factor depends on the represented objects. A scale factor 2 can express that object B is two times faster than A. It can also express that A and B move the same distance but in differently scaled coordinate systems, i.e. the coordinate system of B is two times larger than the coordinate system of A.

Rotating motion:
To map equally motions into different direction, motion paths can be rotated. So, if the *Guide* moves along a line of 100 steps, the *Fellows* move along **another** line of 100 steps. The angle between the two lines is expressed as the rotation factor. A rotation factor of 180 degrees defines that *Guide* and *Fellow* objects move exactly in opposite directions – if the guide moves to the right, the fellows move to the left.
Rotation factors that are multiplies of 90 degree are of special interest. For example, the rotation factor of 270 degrees defines:

- if the *Guide* moves upwards then its *Fellows* move to the left
- if the *Guide* moves to the right then its *Fellows* move upwards
- if the *Guide* moves downwards then its *Fellows* move to the right
- if the *Guide* moves to the left then its *Fellows* move downwards

Delayed motion:
To show that objects can move in similar ways but start at different time points, a delay for the *Fellows* can be defined. This does not affect scale or rotation factors. The motion of the *Fellows* is equally applied with the only difference that the motion starts later.

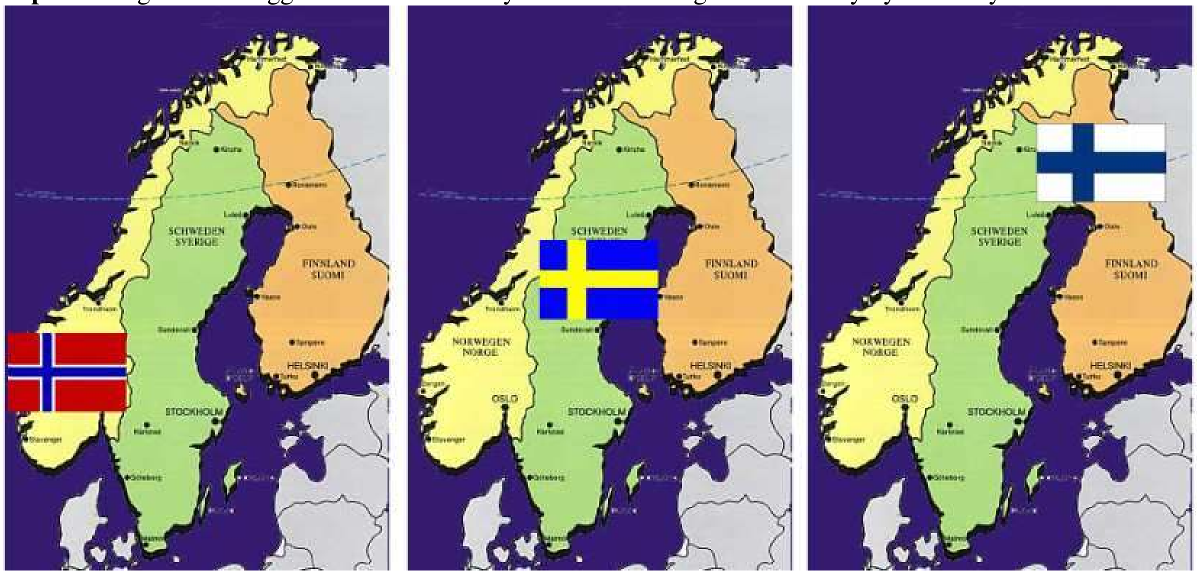
Related Patterns:

Orient to other Objects: Consider the position of *Ruler Objects* to relocate *Ruled Objects* according to geometrically rules, e.g. define that object C should always be in the centre of A and B.
Push Objects: Drag one object to push another object as soon as their bounds touch each other.
Relative Property Change: Define a new position for an object based on its current position.

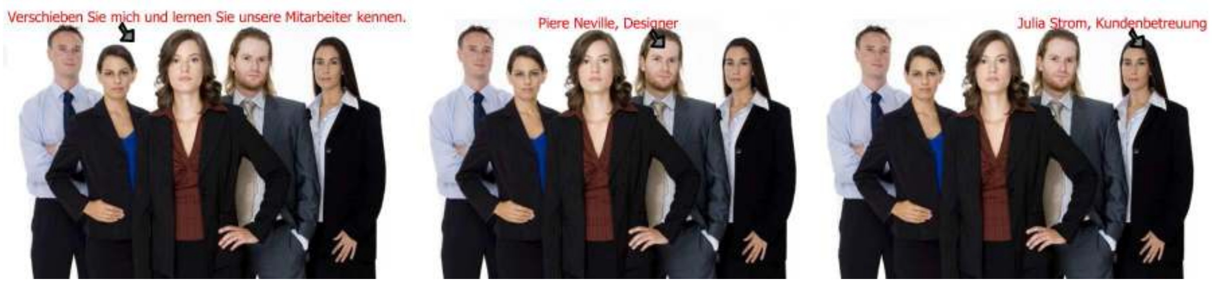
Active Areas (Hot Areas) – (PLoP 2007)

Examples:

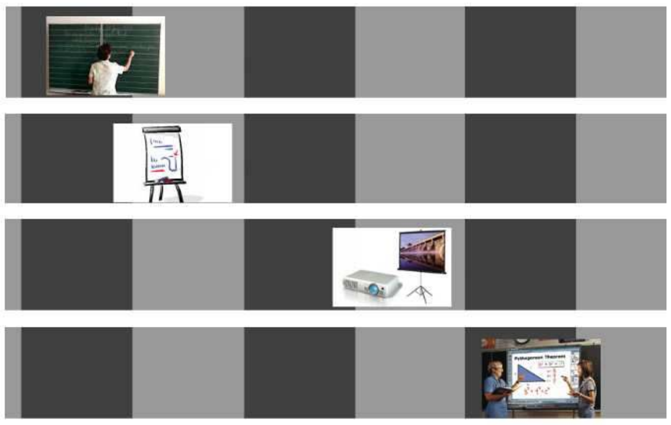
Map: The flag can be dragged over each country. It shows the flag of the country dynamically.



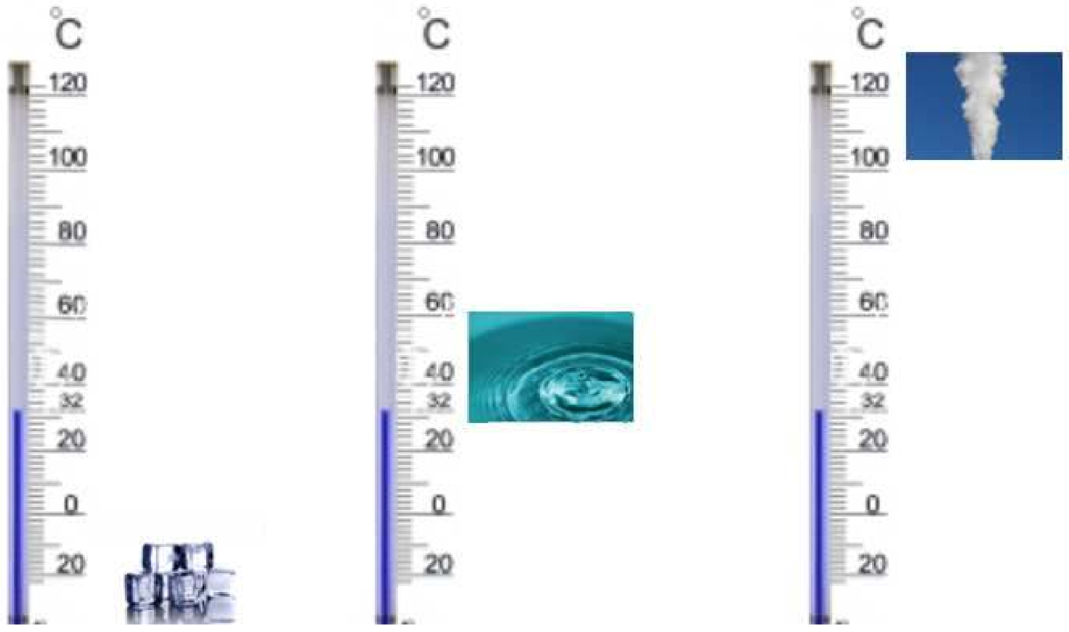
Dragable label: The text can be dragged over each employee and shows his/her name and job description.


Verschieben Sie mich und lernen Sie unsere Mitarbeiter kennen.
Piere Neville, Designer
Julia Strom, Kundenbetreuung

Timeline: The picture can be dragged along a timeline and shows several development states of blackboards and whiteboards.



Water states: the state of water changes according to the temperature.

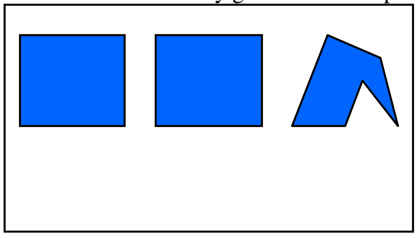


Problem:

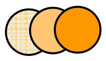
If a graphic consists of several areas where different information items should be provided, e.g. text labels, info boxes or bubbles, it becomes hard to display all information at the same time. Not only is the graphic too packed with information and thus becomes more complex, but the limited space of computer screens sets bounds to the number of information items visible at the same time. It is also true that sometimes only one area should be promoted with additional information, e.g. to control the viewer's attention. Showing all area dependent states at the same time does not show that an object can only be in one state at a time and at which time the state actually changes.

Solution:

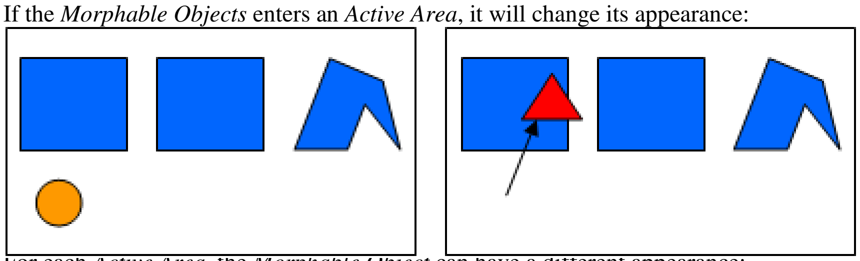
Define *Active Areas* by geometrical shapes:



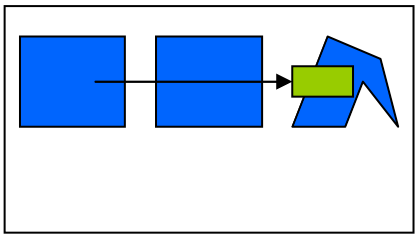
Define a *Morphable Object* that can be dragged:



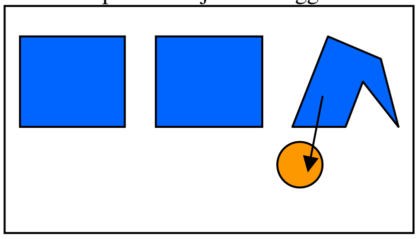
If the *Morphable Objects* enters an *Active Area*, it will change its appearance:



For each *Active Area*, the *Morphable Object* can have a different appearance:

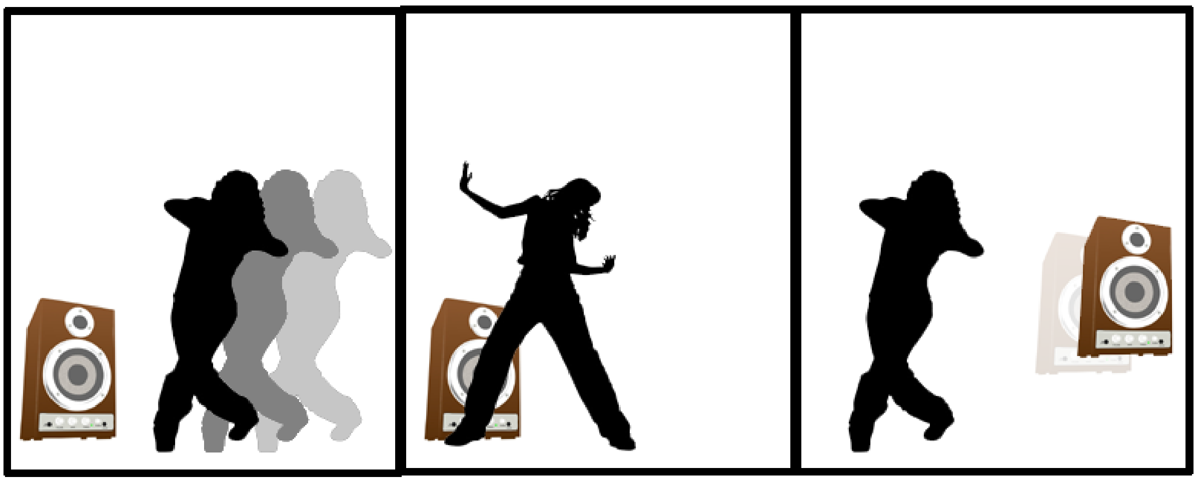


If the Morphable Object is dragged out of all *Active Areas*, it will return to its default appearance:



Details:

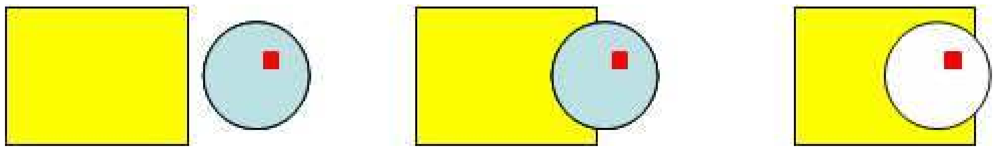
Each *Morphable Object* relates to one or more *Active Areas*. The *Morphable Object* can change its position, for example the users drags it with a mouse pointer. *Active Areas* usually remain at fixed positions at the screens. However, this is not a requirement as the following example shows:



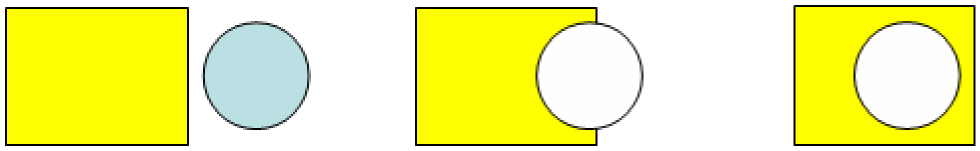
The image shows a girl that either walks or dances (*Morphable Object*). By default the girl walks. If she is dragged over the speaker (*Active Area*) she starts dancing. However, on dragging the speaker away, the girl exits the *Active Area* and stops dancing.

Spatial properties are used to detect at which time a *Morphable Object* enters or exits an *Active Area*. There are three common strategies to distinguish whether an object is inside or outside of another object:

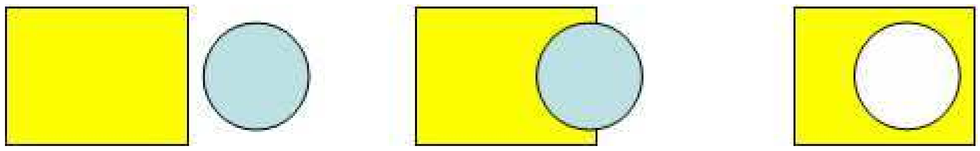
Hot Spot: A single point of the *Morphable Object* is defined as a hot spot. The *Morphable Object* is considered inside of an *Active Area* as soon as the hot spot point is located inside of that *Active Area*. The hot spot is usually positioned in the centre of the *Morphable Object* or in one of its corners. A well-known example for hot spots is the pixel of your mouse pointer that actually triggers mouse clicks or roll-over effects; the hot spot of your mouse pointer is located at the end of the arrow.


The red point is the Hot Spot. If the Hot Spot enters the rectangle then the circle is inside (last figure).

Intersection: The Indicator is considered to be inside of an Active Area as soon as it intersects with it.

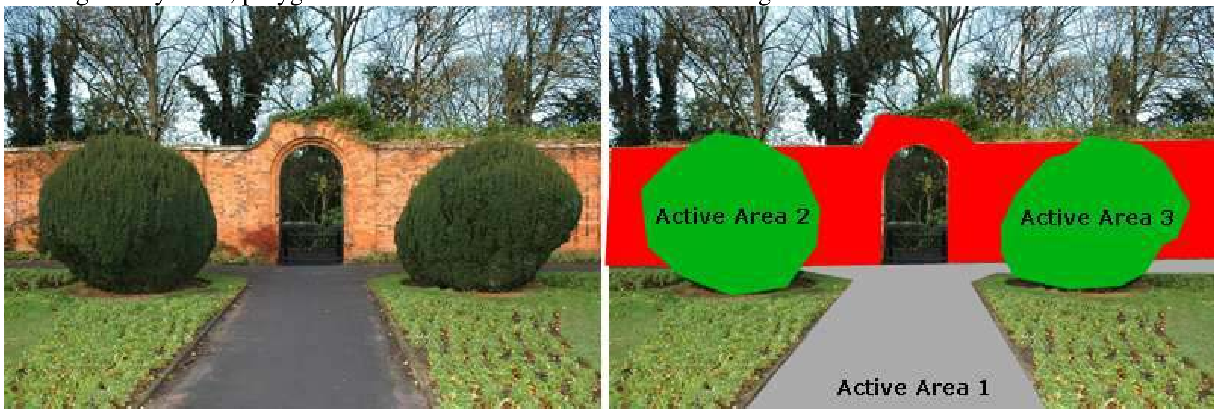

As soon as some points of the circle intersect the rectangle, the circle is inside (figure in the middle).

Inclusion: The *Morphable Object* is considered to be inside as soon as its bounding box is completely inside of the bounding box of the *Active Area.*

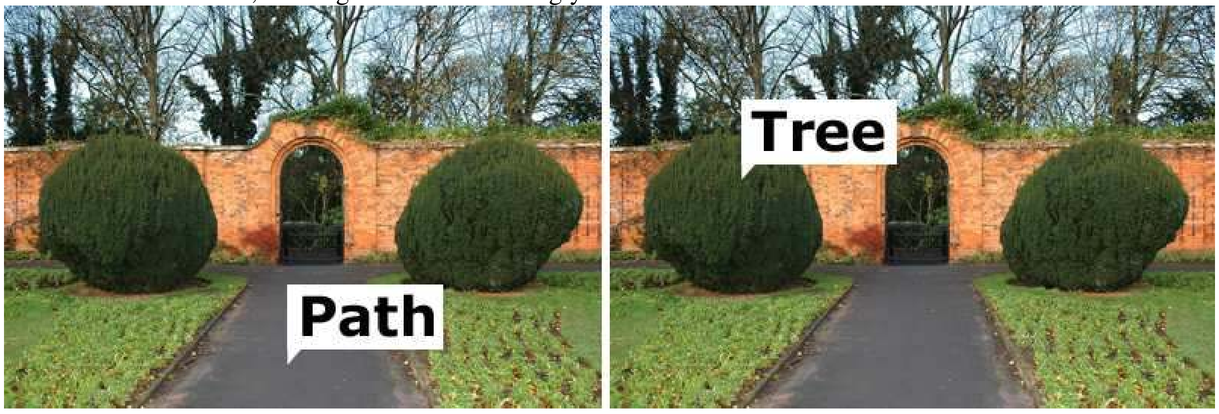

Only if all points of the circle are inside the rectangle, the circle is inside (last figure).

Spatially a *Morphable Object* could be inside of multiple *Active Areas* at the same time. Logically, a *Morphable Object* can only be inside of one *Active Area* because only one visual appearance can be set at a time. To resolve this conflict, one can set priorities for each area. If an element is spatially inside of two or more areas, it is considered logically inside of the area with the highest priority. In most graphical editors the z-order of visual objects (that is which objects appear in front of other objects) provides an implicit prioritization. Areas that are closer to the observer have a higher priority. If intersection is used, another option is available: to indicate that an element is inside of an *Active Area*, the *Morphable Object* can be considered inside of the area that it mostly intersects.

An *Active Area* can be defined by primitive shapes such as rectangles, ellipses, polygons or the bounding box of an image. Very often, polygons are used to define *Active Areas* on images:



Using overlay polygons you can handle different areas within one single image, e.g. a geographical map. After editing the interactive graphic, the polygons are set to transparent . If a text label (*Morphable Object)* is dragged over the invisible areas, it changes its text accordingly:



The example also shows that there may be other supporting objects on the screen that do not affect the behaviour of *Morphable Objects.* The image in the background is the main subject of this interactive graphic, however, it does not play any active part. Instead, the polygons drawn over the image define several *Active Areas.*

There are many ways in which the *Morphable Object* can change its appearance, most commonly are:

- Change the displayed image to show another state or information.
- Use another text to label an area.
- Change the colour, shape or opacity of the *Morphable Object* to highlight that it is over an interesting area.

In general terms, changing the appearance means some visual properties change their values, e.g. file path of an image, level of brightness, colour settings etc. However, it is critical to not change spatial properties that are being used to determine hot spot, inclusion, or intersection. Location, size and rotation are critical properties, accordingly.

What else can I do with this?

- You have a graphic consisting of multiple images and each image should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You have an image with several sub-areas and each area should be labelled individually by a dragable text or symbol that can change its state according to the background.
- You want to use a single object to annotate or label some other graphic elements.
- The appearance of an object depends on its current background information, e.g. different landscapes, regions, data or objects.
- Drag an image along a timeline and show different epochal pictures depending on its position on the timeline.
- Change a displayed image according to its horizontal and vertical position.
- Show the state of an object at different stations.

Rationale:

Showing diverse states according to a context provided by background graphics or spatial position makes cause and effect of object changes more obviously [15]. The visual display of the *Morphable Object* always relates to the nearest background; thus, the gestalt law of proximity is applied [3]. Labels should always be close to the subject of interest [4]. Labels at a different position could distract the user [12]. Another reason to use one but many labels is to direct the attention of the audience and to reduce the complexity a graphic. Reducing the details of a graphic can support the cognitive processing of the content. From a technical point of view the number of simultaneously displayed items has to be reduced because of a low screen resolution.

Related Patterns:

Activator:
The ACTIVATOR pattern is the inverse of this pattern. In both patterns a pair of two states is used for each relation of a dragable element and a fixed area. In ACTIVE AREAS the dragged object changes to an alternative state. In the ACTIVATOR pattern the area in the background changes.

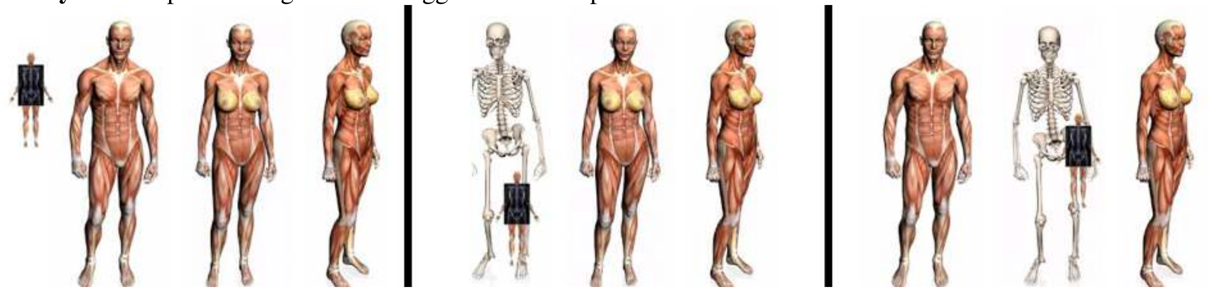
Roll-Over:
Instead of dragging an element around, the mouse pointer directly moves over active areas and causes other objects to change, e.g. labels can switch from invisible to visible or another fixed element changes its appearance. ROLL-OVER images are easier to understand for end-users, however, it is hard to leave the mouse pointer within an area for a longer time and you cannot perform other operation at the same time.

Activator (PLoP 2007)

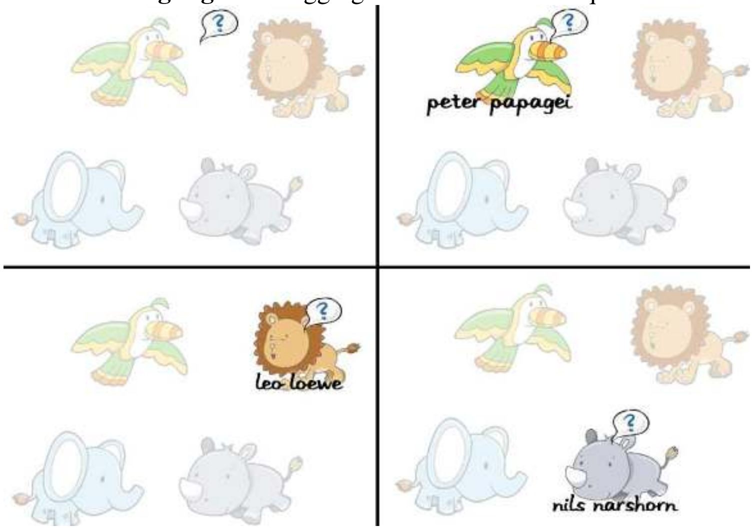
Activate an image or visual object on demand.

Examples:

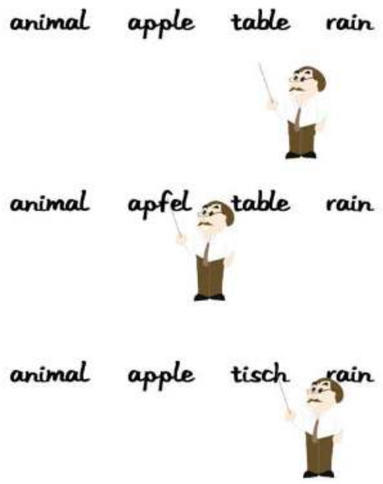
X-Rays: An inspector image can be dragged over each person and the skeleton is shown.



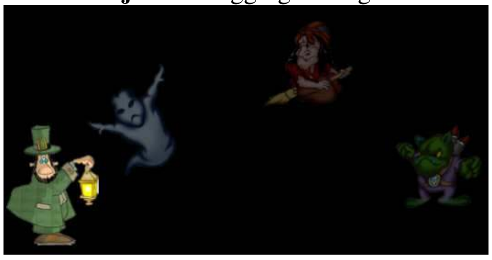
Labels and highlights: Dragging the bubble with the question mark over an animal shows its name.

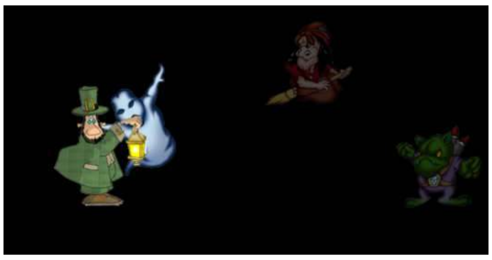


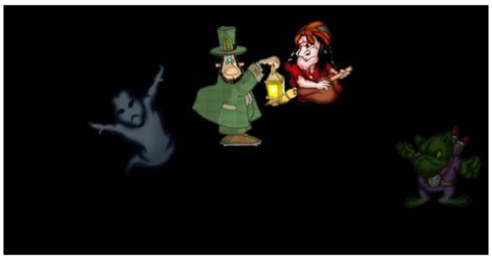
Translate vocabulary: Pointing at a vocabulary translates it.



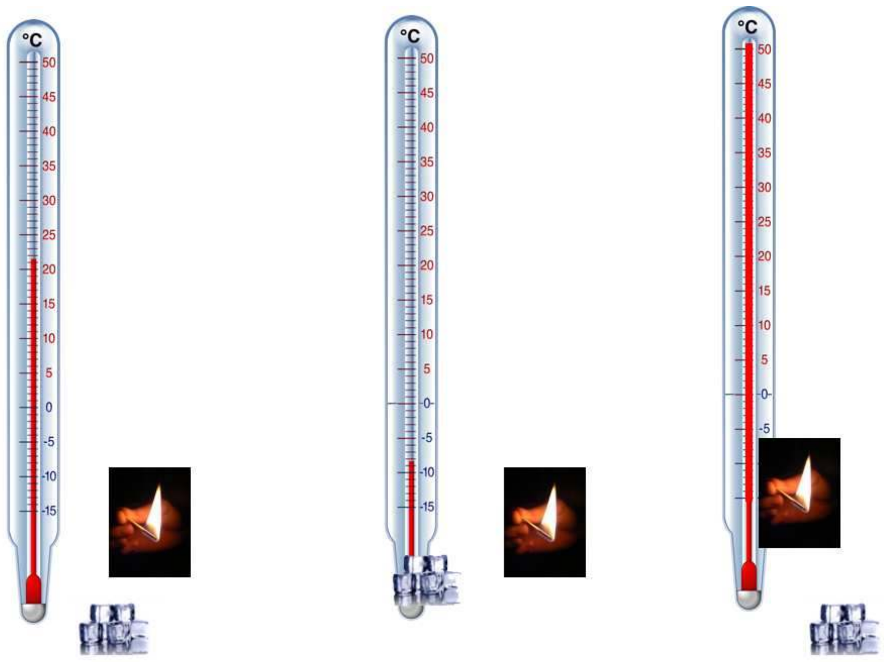
Hidden objects: Dragging the night watchman over the images illuminates them.







Show temperature: The thermometer is an image in the background that can be changed in different ways according to the object that is dragged over it. Ice cubes change the thermometer in a different way than a flame. It is also perceivable that the thermometer changes because a specific type of object is dragged over it.

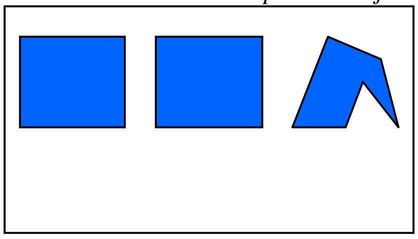


Problem:

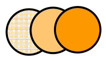
Objects of the real world can be represented visually in more than one way, e.g. you can switch between several states, change the perspective or enrich/reduce the image with information. For objects that can switch between two visual representations there must be an explicit trigger to activate the alternative view. If the user cannot recognize whether or not the displayed image represents the active or inactive state, there must be a visual clue to indicate that an image is activated. Sometimes the mouse pointer can be used to activate an image and switch between two different states. However, a mouse pointer can only indicate the active state for one image at a time. Also, a mouse pointer is a very abstract indicator and does not reveal how or why an object switches to an alternative state. Using the mouse pointer does not add any semantic to a graphic.

Solution:

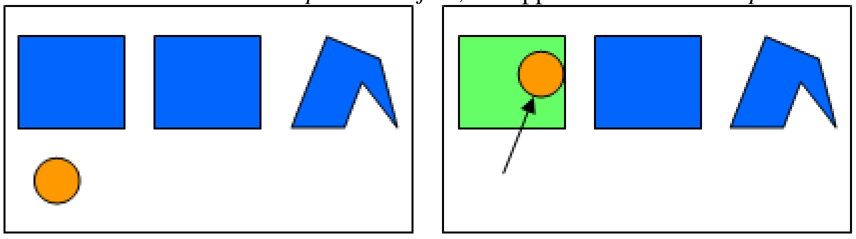
Define one or more *Morphable Objects* which can change their appearance on activation:



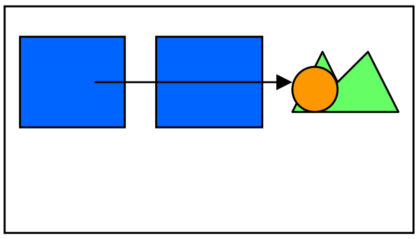
Define an *Activator* that can be dragged:



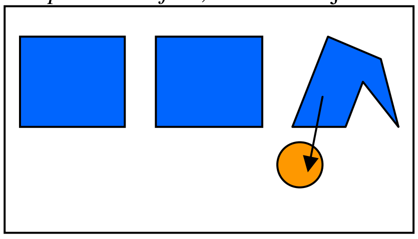
If the Activator enters a *Morphable Object*, the appearance of the *Morphable Object* changes:



If the *Acvtivator* is dragged over another *Morphable Object* it activates that object:



Each *Morphable Object* may change its appearance differently on activation. In general, if the *Activator* exits a *Morphable Object*, then the object becomes inactive and its default appearance is shown:

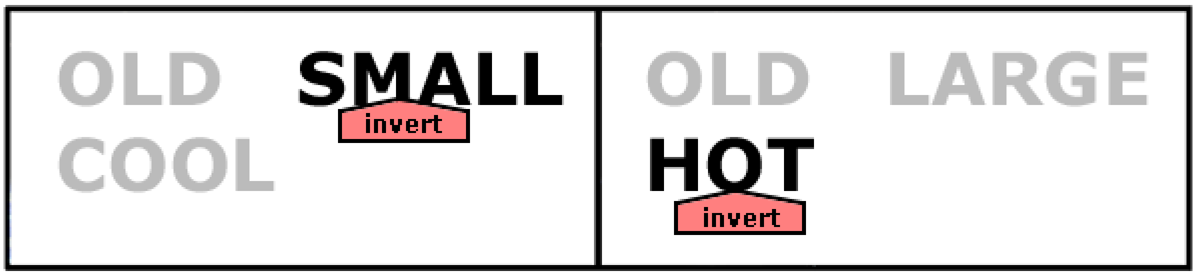


Details:

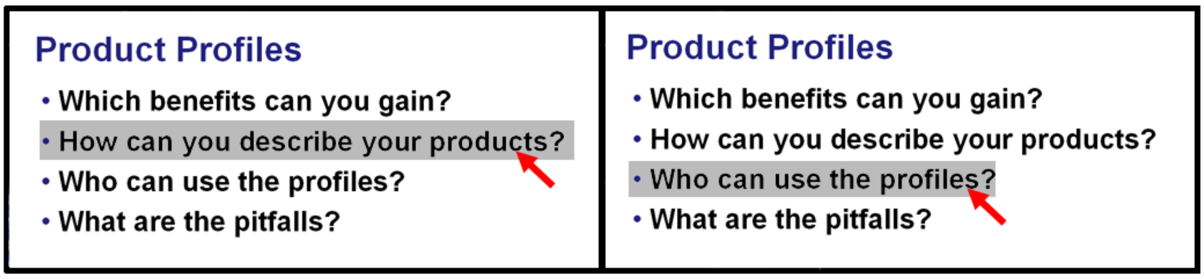
One *Activator* can activate multiple *Morphable Objects*. Activation is triggered on intersection, inclusion or by a hot spot. On activation, visual properties of the *Morphable Object* change. Common examples for property changes are illustrated below.

Image file property. Change the displayed image to show an object in active or inactive state, e.g. a machine that is turned on or off:

Text property. Set another text to change between two different perspectives, i.e. change the language of a phrase, switch from a pro argument to a contra argument, or a verbal description of before and after situations:



Colour property. Change the colour or background colour of an object to direct the viewer's attention to that object:



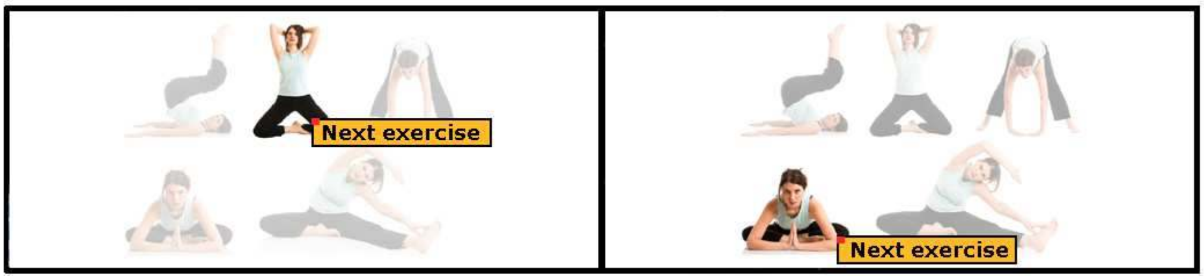
Product Profiles

- Which benefits can you gain?
- How can you describe your products?
- Who can use the profiles?
- What are the pitfalls?

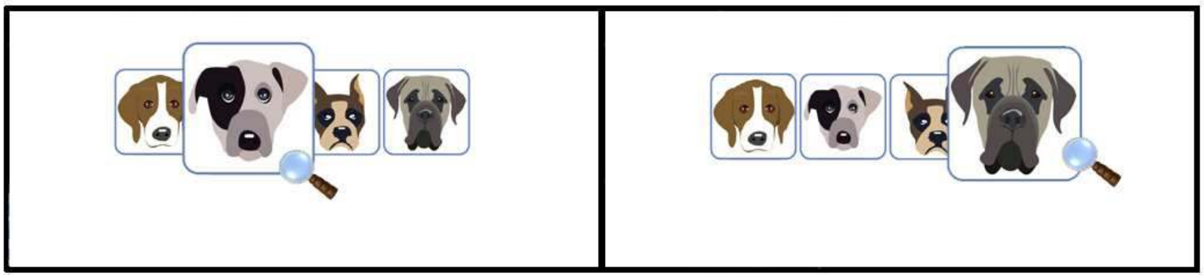
Product Profiles

- Which benefits can you gain?
- How can you describe your products?
- Who can use the profiles?
- What are the pitfalls?

Opacity property. Change the opacity to highlight, to reveal or to hide information temporarily:

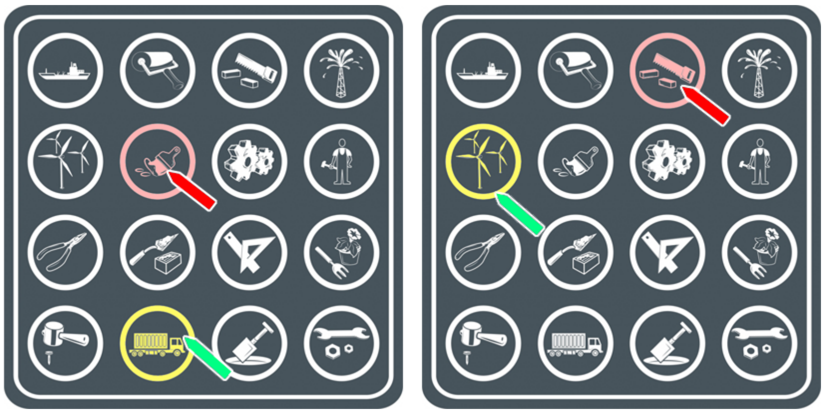


Size and location properties. Set a larger size or relocate an object to let it pop out:



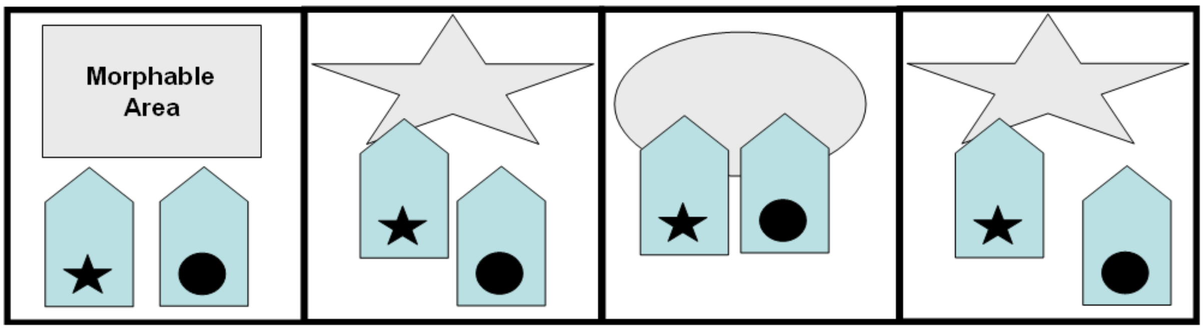
The *Activator* behaves similar to a mouse pointer but introduces some advantages. Mouse rollovers can activate an image only temporarily whereas *Activators* can activate an object permanently. *Activators* can be larger than mouse pointers and thus are better perceived in presentations. Also, an *Activator* can be of any shape and therefore provide additional information about what kind of activation or transformation is performed. In the previous examples the *Activator* "Next exercise" tells the user why the specific image is highlighted, and the magnifying glass indicates that it can be used to enlarge images.

With a mouse pointer only one object can be activated at a time. With *Activators* one can multiply the number of pointers as shown in the next example:



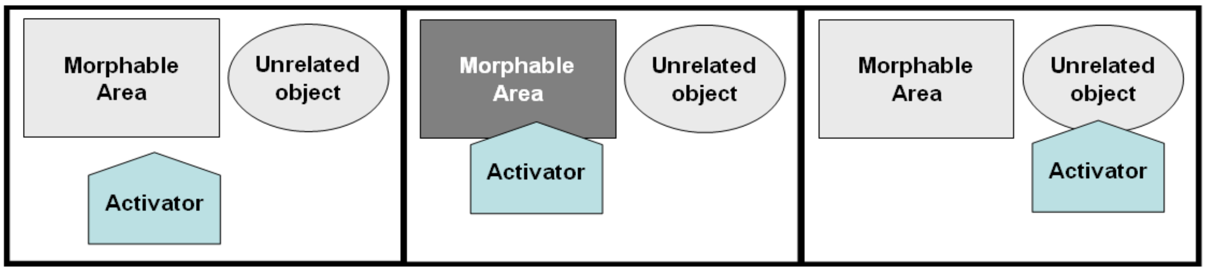
There are two pointers that can be dragged over an image icon to highlight it. The pointers can remain over the images while the mouse pointer can move independently to do other things. The two pointers highlight the image in different ways as well. Used in a presentation, the presenter can refer to differently highlighted images.

However, if there are two or more *Activators* that can activate the same *Morphable Object* then we are running into a conflict. Which activation wins? Or, does the activation by two *Activators* simultaneously trigger a special visual state for the *Morphable Object?* Dragging two different molecules (*Activators)* into a substance (*Morphable Object*) could cause a specific chemical reaction – meaning that the *Morphable Object* shows different results depending on the input of multiple *Activators*. In that case we would have to define special visual states for the *Morphable Object* for all possible combination of *Activators*, i.e. with n *Activators* one has to define 2^n states. To simplify the behaviour of *Activators* it is recommended to allow only one activation at a time. If multiple *Activators* point at a *Morphable Object* the latest activation should be prioritized:



The star arrow (*Activator 1)* first activates the *Morphable Object*. Then, the ellipse arrow (*Activator 2*) activates the *Morphable Object* additionally. Because the latest activation is prioritized, the *Morphable Object* shows an ellipse. On dragging the ellipse arrow out of the *Morphable Object* the star arrow is still activating the *Morphable Object*.

The safest way to avoid confusion is to not use multiple *Activators* for the same *Morphable Object*. Using multiple *Activators* for different sets of *Morphable Objects* does not cause such problems. Objects that are not related to a specific *Activator* do not affect the behaviour.



The *Activator* only changes the visual appearance of the *Morphable Object*. On dragging the *Activator* over the unrelated object nothing happens.

A drawback of *Activators* as compared to mouse pointers is that they introduce an additional level of indirectness. While mouse pointers may activate and deactivate images implicitly and very intuitive, an *Activator* must be dragged explicitly. Because the mouse pointer first has to be moved to the *Activator* to drag it, additional workload is involved. Since *Activators* can come in any shape, a visual affordance that indicates its function is recommended.

What else can I do with this?

- Activate or highlight one of a set of objects. Provide an explicit pointer to the highlighted object.
- Direct the viewer's attention to a specific object on the screen by using a permanent pointer.
- Change the state of an object as soon as it gets in contact with another object.
- Switch between outer and inner view of an object by dragging an "inspector" image over it.
- Hide one representation of an object unless it is explicitly uncovered.
- Map one representation A to another representation B on demand.
- Turn around playing cards or note cards.
- Show a question and reveal the solution by activating the item.
- Translate a text item by pointing on the text.
- Switch between text and image representation.
- Switch between before-after states by using a "transformer" image.

Rationale:

Pointing at objects is a very important tool to direct attention [19]. Activators can point permanently at objects and provide additional semantic. Focussing single objects out of a group helps to concentrate on details. Setting and showing visual states on demand can be helpful for both discovering and demonstrating different views of one object.

Related Patterns

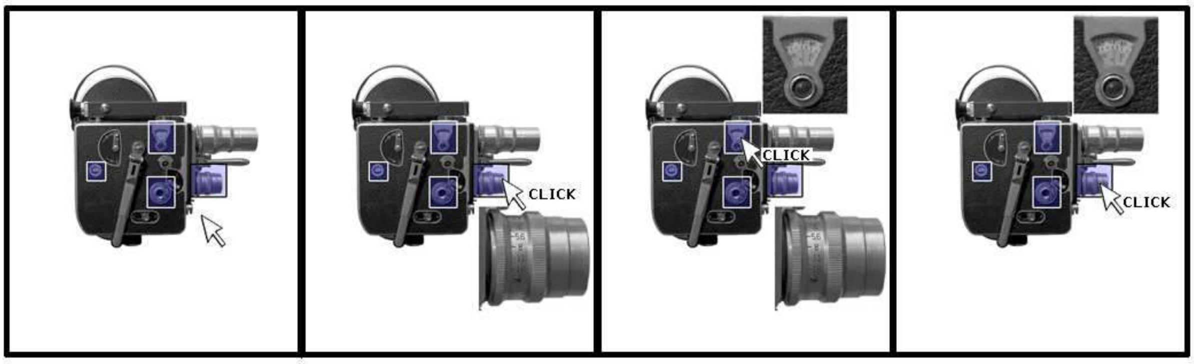
Active Areas: The Active Areas pattern is the inverse of this pattern.
Roll-Over: (see description in the Active Areas pattern).

On/Off Button (EuroPLoP 2008)

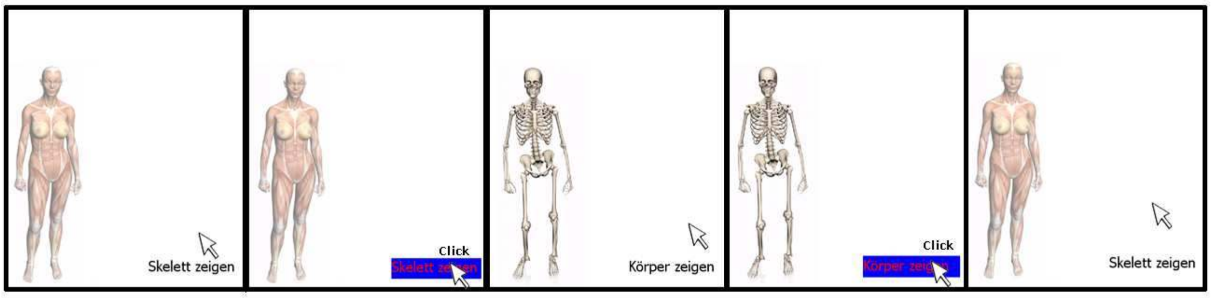
Providing a way to switch between two visual states of an object by clicking at a button.

Examples

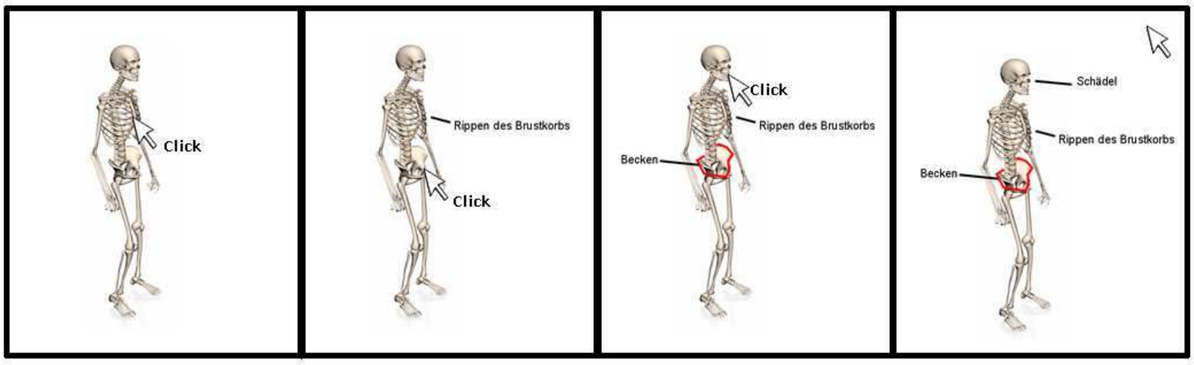
Zoom on Demand: If the user clicks at on of the marked areas, an enlarged illustration pops out. The detailed illustration has to be deactivated explicitly by clicking at the corresponding area for another time.



Switch illustration: The text acts as a button that switches between a medical illustration of muscles or skeleton.



Dynamic labels: By clicking at areas of the skeleton, the labels appear.



Problem

How to add or change information dynamically on the screen and let the user decide at which time which information is present?

Forces

Many subjects require complex graphics to illustrate relations and contextual information but huge graphics can be overwhelming and the observer may not know where to start.

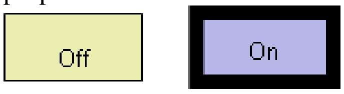
Today's presentation tools allow to add information in a predefined sequence step-by-step, however, depending on the subject it may be desirable to add elements in random order.

Adding information allows to integrate new elements into the existing knowledge representation but what to do with information that is needed only temporarily?

Multiple representation of objects (i.e. to show transformations, causal chains, different viewpoints or levels of detail) can be accessed by navigating to another graphic screen but what if only parts of the graphic should change? How to keep the number of prepared graphics small if different aspects of a graphic can vary independently?

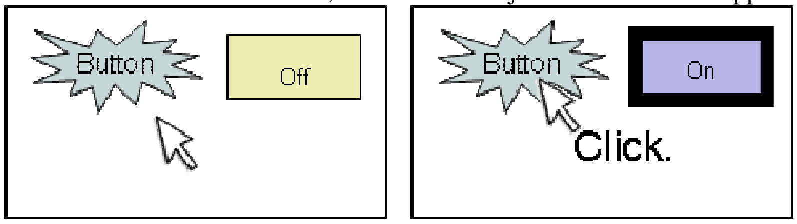
Solution

Use visual objects with two defined states ON and OFF. Each of the state is represented by different visual properties:

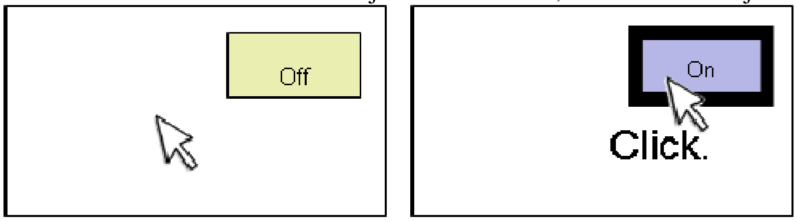


Define an area on the screen as a button:

If the user clicks on the button, the ON-OFF object switches to the opposite state:

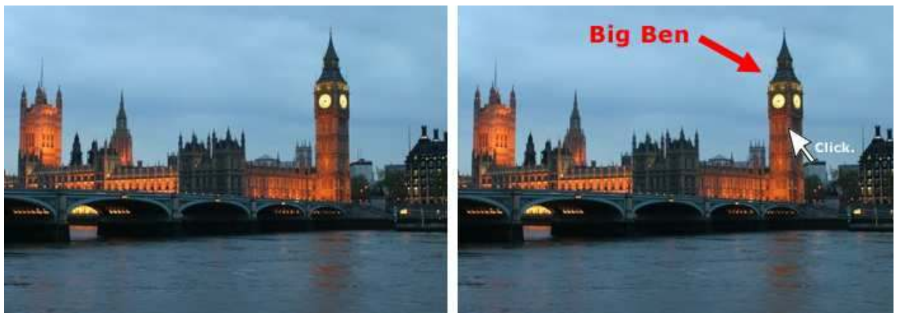


The button can be the ON-OFF object itself. That is, a click on the object will change its own visual state:

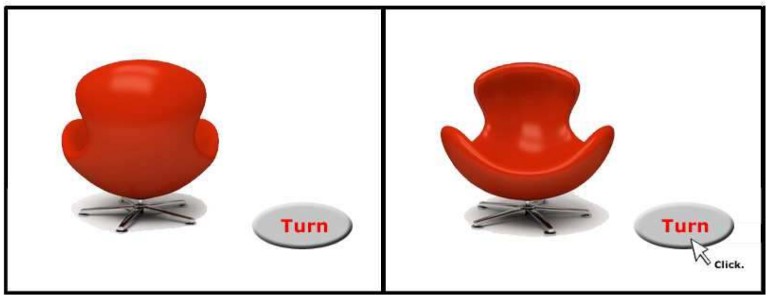


Details

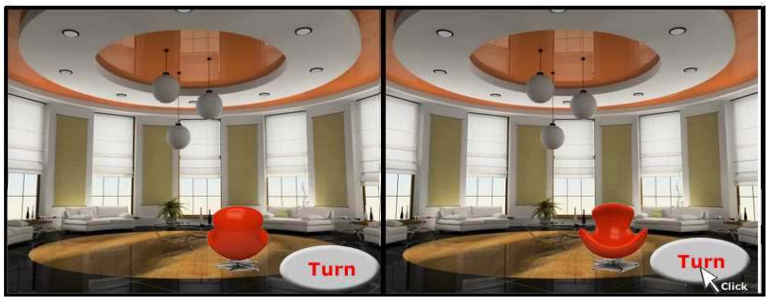
Objects that change their state between visible and invisible overlay an image in the background if they become visible. In this case extra information is added to a base graphic. In the following example a transparent button area overlays the Big Ben. If the user clicks at it, arrow and annotation overlay the picture in the background without changing the picture:



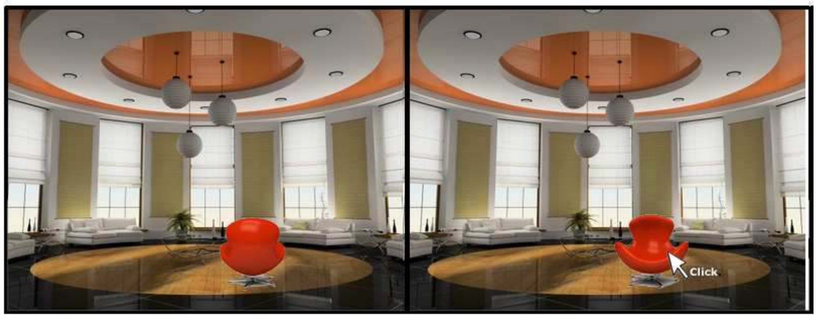
Instead of adding information, buttons can be used to change information. In the next example the "Turn"-button changes the image of the chair:



Of course, buttons can be integrated in a larger picture or illustration. By splitting up an image into independent components, each component can switch between ON/OFF states if required. The next example shows the chair as a changeable part of a picture:

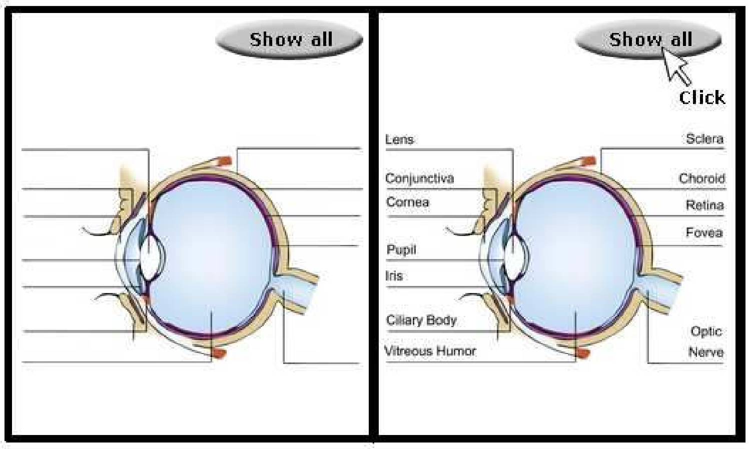


Instead of clicking an extra "Turn" button, the component could be buttons itself:

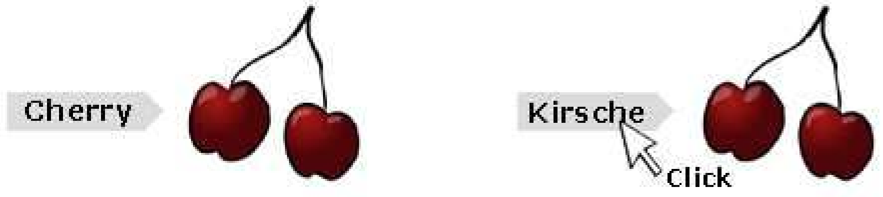


The great advantage of ON/OFF buttons is that the user has full control over the activation of elements. The order of adding or changing information is not predefined. Also, the user is free to choose how long an element remains in a chosen state. He can activate any selection of ON/OFF elements whenever he wants and set the complexity of the shown information as he wish. Information can be filtered and the combination of activated elements is not preset.

An ON/OFF button can turn multiple elements to ON or OFF in one step:



Other properties that typical change are colour or text. Colour changes can be used to highlight some elements out of a group. Text changes can be used to give different information, e.g. translation of a vocabulary:



What else can I do?

- Switch on and off additional information on the screen.
- Show pop-up information in small boxes on demand.
- Show the front and the back of an object.
- Switch on and off overlay information for images.
- Switch on and off labels that explain parts of an image.
- Ask a question and show the answer by clicking at the object.
- Reduce complexity by hiding information if not needed.
- Adding information step by step in random order.

Rationale

Adding visual elements on demand reduces the complexity as only the objects currently needed are displayed. Also, it allows to build up complex graphics step-by-step and helps to avoid cognitive overload by chunking information [CS91] . Having the user decide which elements should be turned on or off at any time gives maximum control to the user. In general, buttons are graphic user interface elements that can trigger action and display states as well [Tid05]. In the context of educational graphics they switch between two states that represent objects. Thus, all objects of the world that may be represented in two major states, can be simulated by an ON/OFF button. Switching the states explicitly by a mouse click is a form of direct manipulation [Shn87].

Drawbacks

Each button can only set two states, ON and OFF. Many objects in the real world, however, offer more than two interesting visual states. To assign multiple visuals states to an object, an INFORMATION DISPLAY can be used.

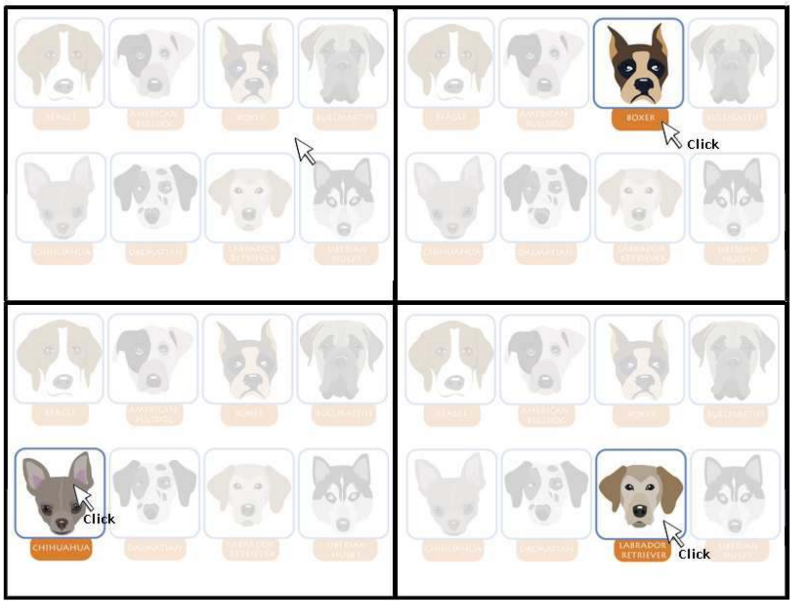
The user has to explicitly turn off an element to deactivate it. In particular, if too many elements are activated simultaneously, an image can become easily overloaded. RADIO BUTTONS and ROLLOVER buttons automatically turn OFF an activated element for the price of having only one element activated at a time.

The changeable graphic elements have to be equipped with their own affordances in such a way that the user can understand where s/he can click.

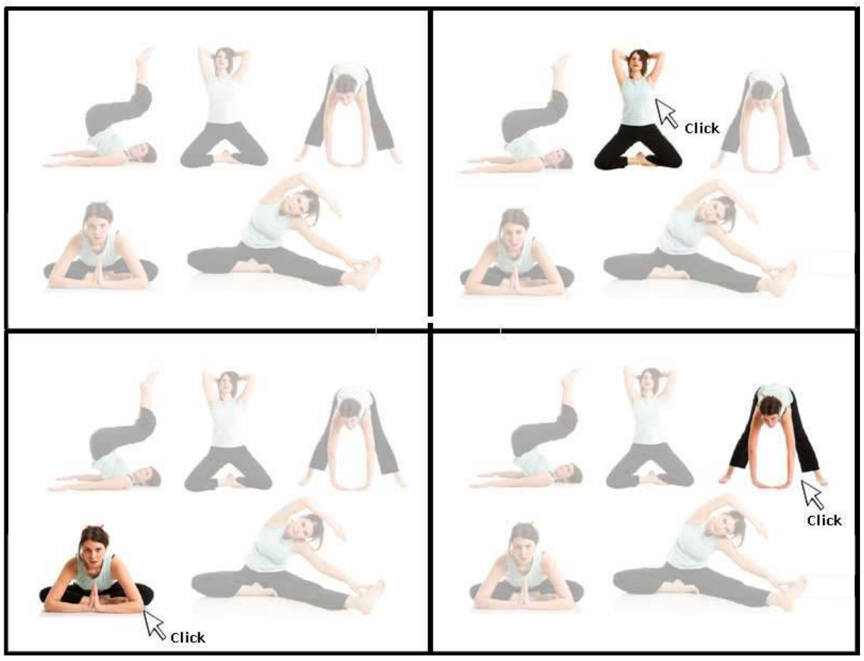
Radio Button (EuroPLoP 2008)

Providing a way to highlight one object out of a group.

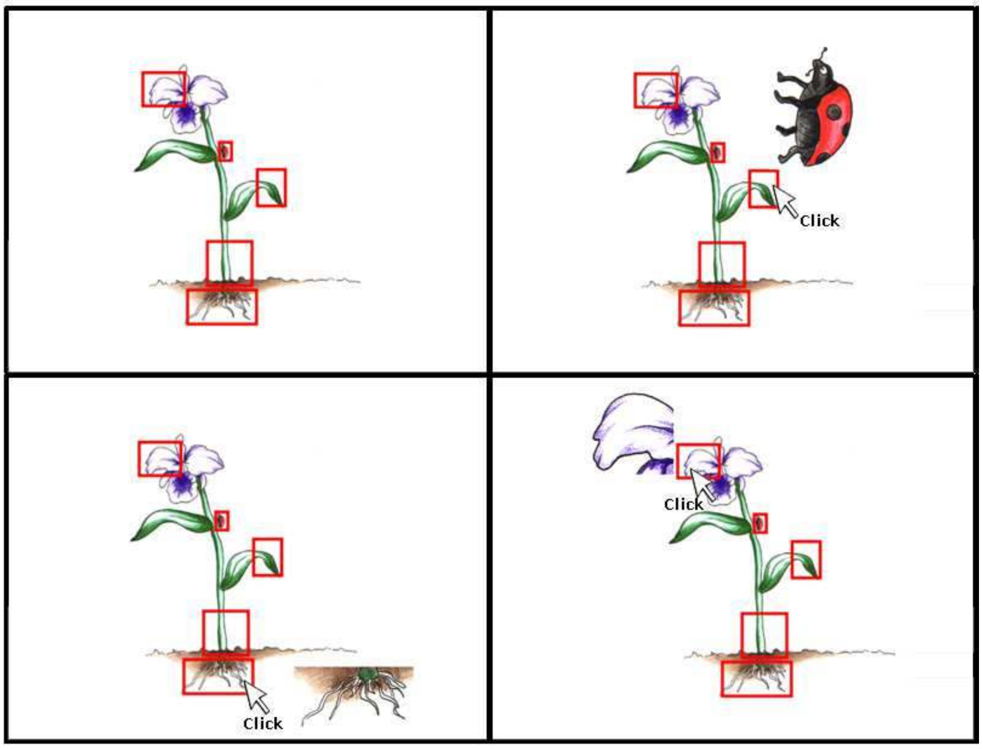
Focus on information. By clicking at one of the dogs, its opacity changes and the user (or an audience) can focus on that information unit. Only one of the information units is highlighted at a time.



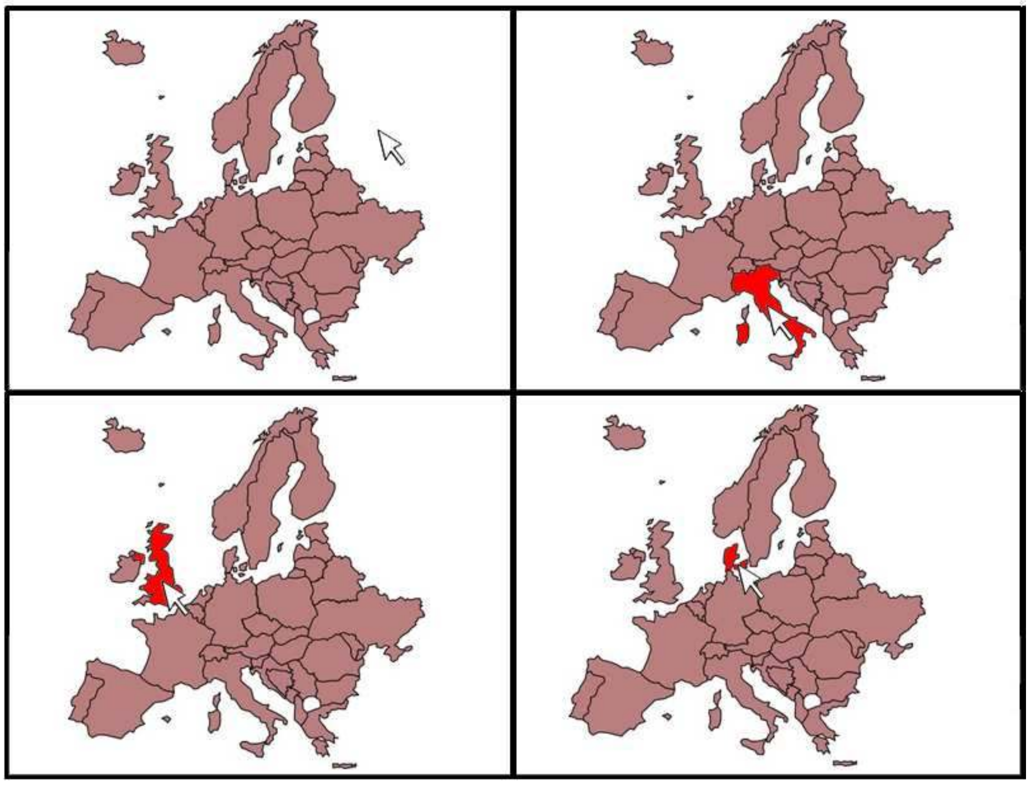
Yoga exercises: One of the exercises is highlighted to show the participants which exercise should be performed next.



Zoom on Demand: If the user clicks at on of the marked areas, an enlarged illustration pops out. Any enlarged illustration shown before disappears. Thus, only one zoom is visible at a time and the attention is directed to only one zoomed detail.



Showing bounds: If the user clicks at any country, the complete area of that country is highlighted. Thus, one can see exactly which parts belong to a country. In particular, this is helpful if the areas are separated.



Problem

How to ensure that only one object out of a group is activated and shows a special visual state?

Forces

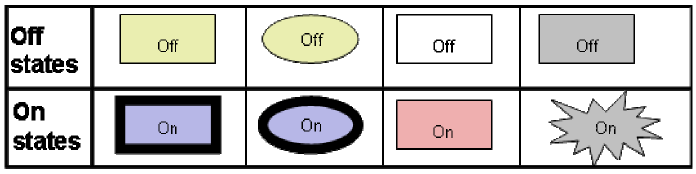
Adding and changing graphic elements on demand makes an illustration adaptive and flexible but can also mess up the screen or hide important information.

Turning graphic elements on and off allows to select and filter the information displayed on the screen but how can one element be selected as a special one? How to focus and highlight one element instead of treating all elements equally?

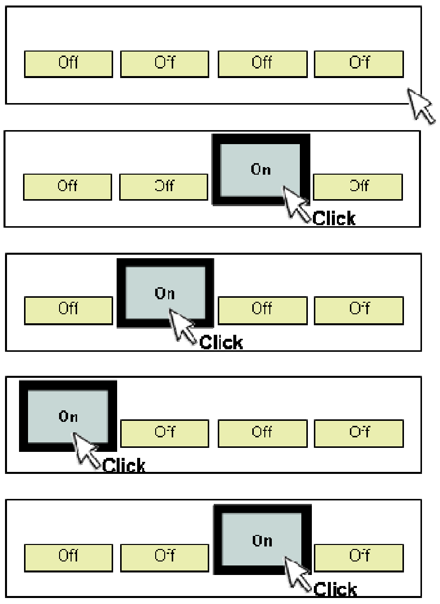
In some graphics one has to avoid that multiple elements are selected simultaneously but users may forget to deselect other elements. Deselecting an element may be inconvenient as it involves extra mouse moves. To deselect an element, the user has to search for the currently activated element on the screen but graphic elements may not provide explicit visual hints which element is currently activated.

Solution

For a group of button objects define different visual states ON and OFF for each button by varying visual properties:

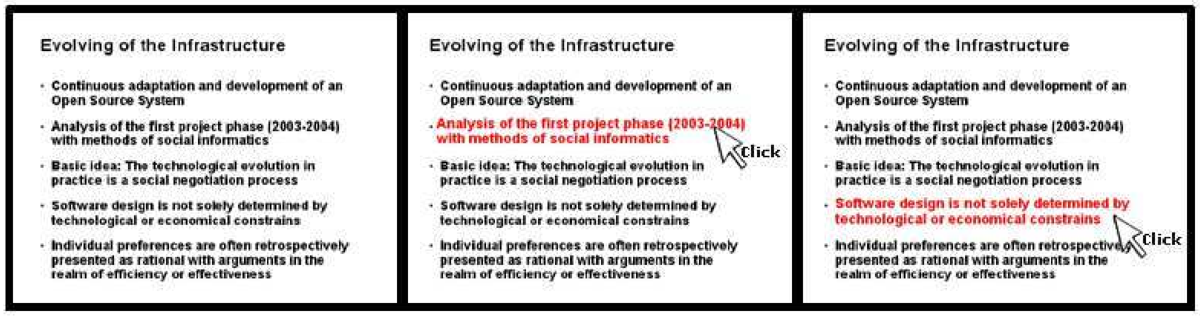


If the user clicks at any of the buttons, the clicked button switches to ON state and all other buttons switch to OFF state at the same time:



Details

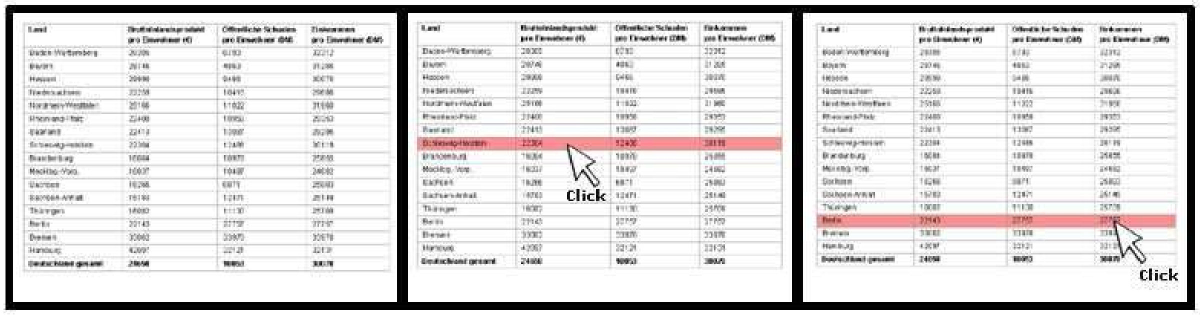
The main purpose of a radio button is to select or activate only one object out of a group at the same time. One can select a single object to focus on it and direct the attention to that object. An object that is highlighted that way stands out and all other objects of the group fade out – either completely or partly. By directing the attention, one reduces the complexity of the graphic because the observer has not to care about all parts at the same time. Also, it is indicated directly which object or item is currently talked about. In particular, this can help in presentations. One can highlight the bullet item or the row of a table one currently talks about:



Evolving of the Infrastructure

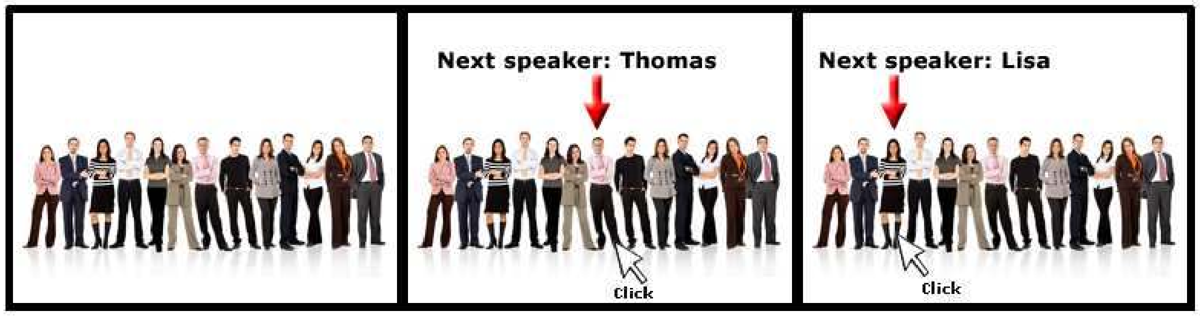
- Continuous adaptation and development of an Open Source System
- Analysis of the first project phase (2003-2004) with methods of social informatics
- Basic idea: The technological evolution in practice is a social negotiation process
- Software design is not solely determined by technological or economical constrains
- Individual preferences are often retrospectively presented as rational with arguments in the realm of efficiency or effectiveness

Click
Click



Land	Bruttoinlandsprodukt pro Einwohner (€)	Öffentliche Schulden pro Einwohner (€/€)	Einkommen pro Einwohner (€/€)
Baden-Württemberg	29305	6792	32212
Bayern	29746	4063	31208
Hessen	29998	6468	30078
Niedersachsen	22259	10419	29588
Nordrhein-Westfalen	25166	11632	31980
Rheinland-Pfalz	22400	10950	29353
Saarland	22413	13087	29296
Schleswig-Holstein	22364	12436	30118
Brandenburg	16084	10870	25658
Mecklenburg-Vorp.	16037	10497	24682
Sachsen	16288	6871	25693
Sachsen-Anhalt	16193	12471	28148
Thüringen	16082	11130	25708
Berlin	23143	27757	37797
Bremen	33082	33975	33578
Hamburg	42097	22121	32131
Deutschland gesamt	**24658**	**10853**	**30078**

Whenever an illustration shows a situation in which only one object can be active concurrently, a radio button is very useful. For example, one can only listen to one radio station at a time (that's where the name "radio button" comes from), only perform one exercise at a time, or allocate scarce resources to only one object at a time. The next example shows how a radio button indicates who is the next speaker in a meeting:



By turning the last activated object automatically OFF, the user input is minimized and the shown information becomes not overloaded.

Once an object is selected to pop out, it remains permanently in ON state until the next object of the group is activated. Thus, the mouse pointer is only needed once to activate an object and can be used for different tasks thereafter (in opposition to ROLLOVER BUTTONS which stress the mouse button for the time it activates an object).

If the user clicks at a button that is already in ON state there are two variations to react: the button may either remain in ON state (meaning there is always one of the objects activated) or it switches to OFF state (meaning that all objects are in OFF state).

At the beginning one of the buttons can be set to ON by default. Another option is to have all buttons set to OFF at the beginning. Note that the visual properties that change can be varied individually for each button. For example, one button could change its opacity while the other button changes its colour.

What else can I do?

- Highlight or select one object of a group.
- Direct attention to a particular object.
- Show explicitly that for a group of objects only one can be active or activated concurrently.
- Show what to do next or who is in the row next.
- Indicate which object is currently focussed on and handled in a presentation.
- Reduce complexity by hiding or fading all information objects that are currently not needed.
- Focus on objects step by step in random access.

Rationale

A radio button selects one object out of a group and provides and implicit focus [Thi90]. It is taken care of that only one object is highlighted at a time. Thus, accidentally activating multiple objects is prohibited. This is important if the graphic is used to show that only one element should be focussed on or represents a special state. The activation state is automatically transferred between the button group. Highlighting or focussing an object directs attention and clarifies which object is talked about. Radio buttons allow to have all buttons simultaneously visible while assigning one special visual state to one of the buttons. Thus, information can be available at all times but it is assigned with different priorities.

Drawbacks

Each button can only represent two visual states. There is no explicit indicator which element is currently selected because the mouse pointer can move to other positions while the selected element remains in ON state. Thus, the graphic element has to provide its own visual hint that it is activated.

The graphic has to provide affordances to point out which areas are clickable in order to activate one of the radio buttons. These affordances may add an undesirable interference with the graphic itself. Radio buttons are often used to reduce the number of visual elements of a graphic. Hence, if new visual elements are required to activate the buttons, the benefits are nullified. For this reason, radio buttons work best if a base graphic is in use that offers implicitly such affordances, e.g. the bounds of a map or the spatial parts of objects or charts.

Synchronize Object Motion (EuroPloP 2010)

Show exactly how the motions of objects depend on each other.

<u>**Context**</u>
To illustrate the motion of objects and their causal relationships it is often needed to move several objects at the same time and to have their motion orchestrated, e.g. two objects move exactly in the same way, simultaneously in opposite directions or follow the same paths.

<u>**Problem**</u>
A user can control only one object at a time. Even if multiple objects are moved user-controlled at the same time, it is impossible to direct all objects in such a way that the motion paths are identical. Predefining several animation paths for the different objects gets in the way of spontaneous interaction with the graphic.

In fact, predefined paths will lead to deterministic animation that is not interactive at all. Thus, experimenting and exploring what happens if one object is moved would not be possible. Defining animations paths is very time consuming as well and it forbids the ad-hoc creation of new animations and take control of the interactive graphic.

Simply grouping the objects that should move together could only help if all objects are supposed to move in exactly the same way. Such large groups often complicate the access to other objects on the screen and you accept that each element of the group can cause motion. Often, there are implicit motion relations between objects but these do only apply if one of the objects is explicitly set into motion. So, what triggers the motion of specific objects? How can you reveal motion relations between objects? How do you handle variations in speed, directions or time delays? What can you do if the objects that are supposed to move are on backgrounds that use different scales?

<u>**Solution**</u>
Define *Guide* and *Fellow* objects that synchronize their motion automatically. The motion of the *Guide* object is exactly captured in a motion vector. The motion vector is then applied to all related *Fellow* objects either immediately or delayed. Also, the motion can be transformed, i.e. scaled or rotated.

Scaling motion:
To respect different scales and speeds of objects, the motion vector of the *Guide* and its *Fellows* can be scaled by a scale factor. The scale factor defines the relative step size between object A's and B's motion. For example, let the scale factor be 2 and object A moves 15 pixels. Then object B moves automatically 30 pixels. On the other hand, if B moves 50 pixels, then A moves 25 pixels.

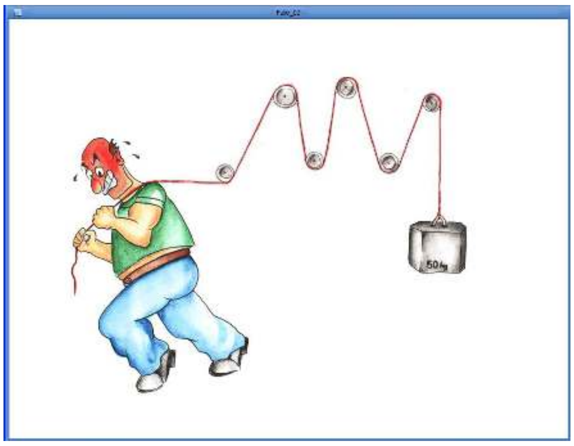
Rotating motion:
To map equally motions into different direction, motion paths can be rotated. So, if the *Guide* moves along a line of 100 steps, the *Fellows* move along **another** line of 100 steps. The angle between the two lines is expressed as the rotation factor. A rotation factor of 180 degrees defines that *Guide* and *Fellow* objects move exactly in opposite directions – if the guide moves to the right, the fellows move to the left.
Rotation factors that are multiplies of 90 degree are of special interest. For example, the rotation factor of 270 degrees defines:
- if the *Guide* moves upwards then its *Fellows* move to the left
- if the *Guide* moves to the right then its *Fellows* move upwards
- if the *Guide* moves downwards then its *Fellows* move to the right
- if the *Guide* moves to the left then its *Fellows* move downwards

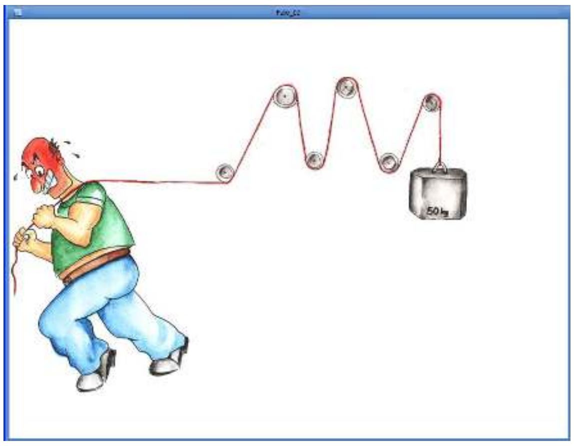
Delayed motion:
To show that objects can move in similar ways but start at different time points, a delay for the *Fellows* can be defined. This does not affect scale or rotation factors. The motion of the *Fellows* is equally applied with the only difference that the motion starts later.

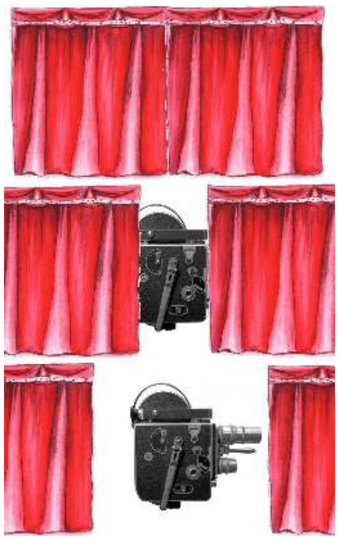
Examples

- Objects should move in the exact same way.
- Objects move in a similar way.
- Object motions relate to each other.
- Show cause and effect of object motions.
- Show that one object drags another object.
- Use a small object to navigate a large object.
- You need sliders or scroll bars to position objects.
- To compare relative speed between objects.

Demonstration of a lifting block. The user pulls a person horizontally and the block moves vertically in synchronisation.







Hiding objects. A curtain is opened and reveals some hidden content. You can drag at each side of the curtain horizontally.

Zoom-Navigator. The red rectangle selects the part of the zoom image that is displayed. To do so the zoomed image moves into the opposite direction when the rectangle is dragged. Because it is three times larger, it moves three times faster.

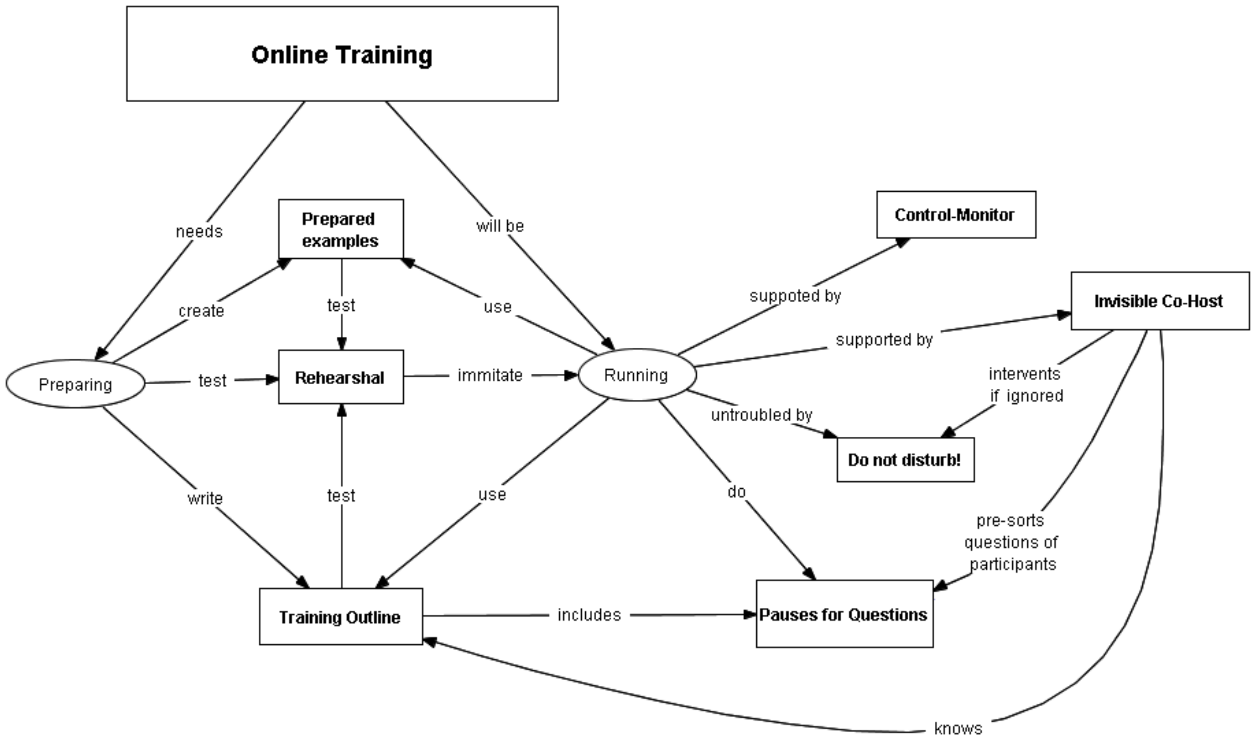
Appendix F

A Pattern Language for Online Trainings

Abstract
This is a small pattern language for running online trainings. The language starts with an entry pattern ONLINE TRAINING. All other patterns are supporting this pattern and help to make online trainings more alive. Some of the patterns are not exclusive for online uses, for instance PREPARED EXAMPLE or TEST RUN. These patterns could be part of a pattern language for trainings as well. They are included in this collection because they are important for online settings particularly. Another example is PAUSE FOR QUESTION. This should be applied in classroom education as well. But the present pattern description takes special forces into account, i.e. online you have to plan *more* pauses due to the lack of any non-verbal feedback.

Mining Ground
The patterns ground in experiences of online training events of the German information portal e-teaching.org. Since spring 2006 there have been 14 online trainings, hosted by the author and guest trainers. There are usually between 30-50 participants attending the live event. Each event is recorded and can be accessed later. The trainings are open educational resources and can be accessed at[36]: **http://www.e-teaching.org/community/communityevents/schulung/**

Besides our own experiences, we discussed best practices of using the conference system Adobe Connect at a user meeting. Many of the patterns we applied have been in use by others independently, i.e. Invisible CO-HOST, CONTROL MONITOR and of course TEST RUNS. Though we have gathered our experiences using a particular conferencing system, they are not limited to that system. We have tested other systems as well and their functions are quite comparable. The present patterns should be useful for any of the standard web conference systems.



The pattern map shows the relation between the patterns and how they support each other. The map clarifies that some patterns are applied in advance to prepare for the training while other patterns are more important when the actual live training takes place.

[36] The trainings are in German.

Acknowledgement
The author wants to thank his shepherd Nuno Flores for all his help and very thoughtful suggestions. I enjoyed working with Nuno and I think I have not only learned new ideas for this paper but improved my pattern writing style in general. I appreciate all the time Nuno has invested in the shepherding process! I would also like to thank all people who have provided feedback to this paper and the patterns. Many helpful suggestions from the Writer's Workshop at EuroPLoP 2009 have helped to shape this paper as well. Thanks to all participants.

Online Training

Alias: Webcast, Tool Demonstration

Context
Using software applications is a fundamental activity in everyday work life. This applies to teachers, students and employees in the same way. To acquire the desired level of skills, trainings have shown to be an effective way to impart usage scenarios, process steps, tips and tricks. Trainings are suitable to introduce new software products or versions promptly.

Problem
The number of participants is usually limited for classroom trainings. There is a fair amount of costs for travelling to the training site, a loss of working time, and rooms need to be allocated. The calendars of the trainer and the several participants have to be synchronized which is not always an easy thing to do. If potential participants are located at different sites, finding a time suitable for everyone becomes even harder.

Forces
Time For many software applications, there is a need for sophisticated trainings. However, the time resources of trainers are limited. This is true particularly for part time trainers who have other job activities on their agenda as well.

Availability Training services should be available on-demand and just-in-time, for example if new tasks should be coped with, or a new colleague needs to be introduced to the software.

Overview Sometimes people just want to get an idea of a software product without committing to a time intensive training session.

Costs The investment of time and money is a barrier in particular for trainings that are important but not mandatory (or at least judged to be not mandatory).

Special interests For special interest trainings (e.g. a special field of application for standard software) there are usually not enough attendees at a local site while there might be enough interested people on national or international level.

Feedback Training videos offer no way to ask questions, or to signal difficulties in understanding the demonstrated functions.

Solution
Online trainings allow the participation from anywhere independent of the location. If the online training is recorded, the recorded material can be accessed at anytime.

Details
Today's web conference systems allow broadcasting computer screens and actions to many participants as a web cast. A trainer uses a headset and comments his action on the screen. Participants can use a chat window to give feedback and ask questions. To make life easier for the trainer, and to engage in communication with participants, a CO-HOST is a reliable support. In this case, the training is not led by a single trainer but supported by an assistant who takes care of questions and discussion in the chat.

To check how demonstrated work steps are broadcasted to the participants, a second monitor can work as a CONTROL MONITOR. This monitor shows the broadcasted screen from the perspective of the participants.

Integrate more PAUSES FOR QUESTIONS (“Did you understand everything?“, “Any questions?”) as you would do in your classroom education and use PREPARED EXAMPLES. Plan your trainings to take no longer than 45 minutes (60 minutes being the limit!). People get tired in longer lasting on-line sessions. Shorter sessions, say 15 or 30 minutes, will also do well. While you are hosting an online training you should provide a personal and friendly atmosphere. Don’t forget to smile occasionally even if you cannot see your participants and their reactions.

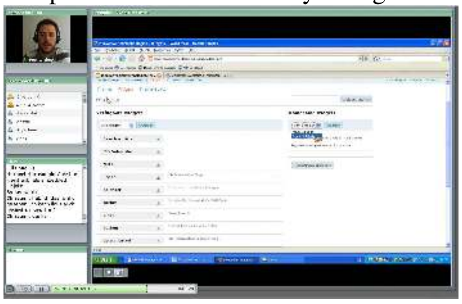
Obstacles
Preparation Gap There is a huge difference between planning and running trainings. Only if you run and implement the training, the real challenges in operating the software and applying meaningful actions will reveal. Whether the instructional outline works can only be tested if you run the training. Therefore, do a REHEARSAL before starting the training with real participants.

Technical Problems According to Murphy’s law, if anything can go wrong, it will! If anything goes wrong during the live event this could cause inconvenient delays and may let the participants quit even before the event started. Furthermore, technical errors look very unprofessional – even if the trainer or host cannot be accounted for the error. Therefore, always check the technology in advance, fix any broken components, and analyze sources of failure.

Benefits
- Participants are not required to travel to a trainings site. This saves time and money. Training sessions can be integrated into the calendar of a regular working day.
- If a participant doesn’t like the training – or realizes that the content does not fit to his learning goals – one can always quit he training without causing any trouble.
- Chat windows enable participants to ask questions during the presentation (an advantage over pure training videos).
- Records allow to watch the training later on and to replay specific sections for a better understanding (an advantage over classroom training).
- No physical rooms need to be allocated or rented.

Liabilities
- Cognitive load for the trainer due to the lack of relaxing pauses.
- No hands-on-practice during the training.
- A lack of non-verbal feedback to see whether participants grasp everything.
- More efforts to organize the technical infrastructure for training.
- A fast internet connection is required. Still, some interruptions of audio or video may occur.
- Delayed broadcast of screen operations may cause an asynchrony of audio comments and screen video.

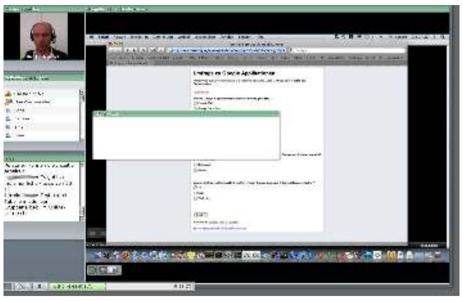
Examples:
The following examples show several online trainings hosted by www.e-teaching.org, each introducing a different software application. Each example was hosted using the same conference system, Adobe Connect. However, there are other tools (see below) which enable to run the same type of event. The trained software applications vary in their screen dynamics.
The content management systems Drupal and Wordpress are examples for web based applications where the complete screen content only changes when a new page is loaded.

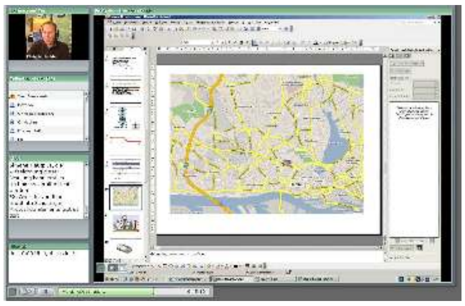

Wordpress training
http://connect.iwm-kmrc.de/p35287287/

Drupal training
http://connect.iwm-kmrc.de/p68096301/

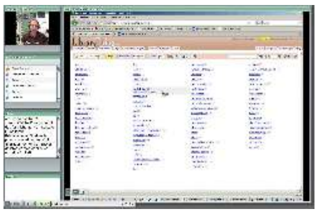
Google Apps is an example where direct manipulation causes more frequent screen changes. In this training the participants where asked to fill out a questionnaire and their answers were evaluated live with Google

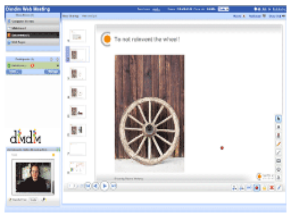
Spreadsheet. The PowerPoint training is an example for frequent screen changes that occur when a slide is edited or an animation starts. While the editing was broadcasted well, the animation was not smooth because some frames were left out.


Introduction to Google Apps
http://connect.iwm-kmrc.de/p27372695/


Instructional Animations in PowerPoint
http://connect.stine.uni-hamburg.de/p65354949/

Tools


Adobe Connect


DimDim

OnSync

Adobe Connect, OnSync, and DimDim are examples for web based meeting systems that only require a flash plug-in. Hence, most web users can attend trainings without installing anything on their computer. Adobe Connect is very simple to handle and supports screen recording on Mac computers. DimDim is based on open source components and offers free hosting for up to 20 participants. OnSync has very good compressions but is less intuitive than Adobe Connect.

Control Monitor

Alias: Second Screen

Context
In an online training the trainer will demonstrate the use of software on a screen that is broadcasted to participants who are located at another site. In opposition to classroom settings he will not see exactly the same image on the screen. Participants will see the screen in their browser window with a delay of time, lower screen resolution and a lower refresh rate. Furthermore, the broadcasted screen is usually integrated into a conference system which brings additional panels on the screen (e.g. video-image of the trainer, chat window).

Problem
The comments of the trainer should refer to what the participants see on their screens – and not what he sees on his screen. If the audio comments do not fit to the screen video shown, there is a danger of inducing false mental models for the participants.

Forces
Asynchronous streams When speaking his comments, the trainer refers to the process steps he currently applies. However, participants receive the broadcasted screen with a time delay (whereas audio might be delivered without delay).

True-to-detail High frequent changes of what is shown on the screen (e.g. animation, videos or rapid input of text and drawings) will not be broadcasted true-to-detail due to the compression and low bandwidths. For examples, some frames may be left out or the viewer only shows the input of a whole word rather than the singular letters in a text field.

Resolution Since the captured screen is scaled down and broadcasted with lower image quality, details may get lost. In an extreme example, the trainer might see things clearly whereas participants see things only blurred.

Lag Frequent switches between windows frames or scrolling the screen will delay the broadcast since every pixel on the screen changes. However, the trainer will not recognise this because his own screen just acts normally.

Solution
Put a second computer and monitor besides the computer that is used to capture the training. On this second computer one can login as a normal participant. Hence, the trainer can see the screen from the perspective of the participants.

Details
The second monitor does not only show the broadcasted screen but all other panels of the conferencing system as well (e.g. his own video image recorded by the webcam). On his first monitor, the environment of the conferencing system is hidden because the trainer operates the software that is trained.

If the trainer sees a delay on the second monitor, he can mute for a few seconds to reduce the generated amount of data. That will release server capacities and bandwidth.

Obstacles
Missing Eye Contact While looking at the second monitor, the trainer no longer looks into the webcam. He no longer has “eye contact” with his audience because he is looking at some corner of the screen. Therefore, the control monitor should be located close to the primary monitor to avoid moving the whole body of the trainer out of the recorded video image. Not looking into the camera can affect people to think that the trainer is inattentively or unconfidently.

Interference However, if the control monitor is put too close to the primary monitor, the perception of two monitors with almost (but not quite) the same content can irritate and distract the trainer.

Echo Effects The speakers of the second computer have to be turned off. Otherwise, the microphone of trainer might record its own audio output recursively and annoying echo effects occur.

Benefits
- By looking at the second monitor, one changes his audio comments. Rather than saying “Now you see,” one says “In a moment you will see”. Hence, the audio comments are more appropriate to what actually happens on the screen.
- A second monitor enables the trainer to observe chat messages and read them in planned PAUSES FOR QUESTIONS.

Liabilities
- Looking too many times to the control monitor interferes with the work flow, for the operations will still be applied on the primary computer and not on the second one.
- Additional hardware is required.
- Time consuming setup of the trainer’s desktop.
- The trainer frequently looks to the side instead of into the camera which can be irritating.

Examples

If the hosting computer uses an unusual screen format or a very high resolution, a second monitor can help to check what the participants see.

Small laptops or Tablet-PCs are perfect as a second screen because they are easy to setup on your desk and don’t take too much space. Also, such small computers usually have a “low” resolution (e.g. 1024x768 pixels) and you can see what happens under “worst” conditions.

Using a second monitor is particularly helpful if there are frequent changes on the screen. In trainings for PowerPoint, Flash and Photoshop, the trainer was frequently checking whether the manipulations were fully captured by the conference system.

In another example, a trainer was not using a control monitor. When he was scrolling quickly up and down on a website (to just show what is at the bottom of the page), he did not realize that no participant ever saw the bottom of the website because it did not last long enough at that position to be captured by the conference system.

Pauses for Questions

Alias: Any questions?

Context
Planning and running an online training to let participants acquire software skills.

Problem
In online settings there is no non-verbal feedback that could indicate whether the demonstrated process steps are understood, whether the pace is appropriate, or whether
interest arouses. Without this feedback the trainer cannot adapt the training to the situation and it gets harder to optimize learning effects.

Forces
Visual Feedback Stereotypical signals such as frowning, staring bored, widened eyes, confirming smiles etc. are giving conscious or unconscious feedback to the trainer, indicating whether his style of training is adequate or needs to be adapted. However, in an online training the trainer does not see the participants.

Signaling It should be allowed to ask questions at any time. However, how can a participant indicate need for more information if he cannot simply raise a hand?

Pace Continuous talking of the trainer is stressing and participants need some relaxing time every now and then to process the new information. However, idling moments have different effects in online trainings and seem to be awkward.

Solution
Plan to invite participants to ask questions (“You understood everything?“, “Any questions so far?”) explicitly and more frequently than in classroom trainings. Announce at the beginning of the training that you will include several pauses in which questions can be answered.

Details
Since there is no implicit feedback from non-verbal communication, we have to ask explicitly for feedback more frequently. At the end of the training, too, the trainer should ask for feedback: “Were the amount of information and duration appropriate?”, “Did you miss any information?”, “What would you like to hear the next time?”, “Was the quality of audio and video sufficient?” Free answers can be typed into the chat window. Some conference systems allow preparing questionnaires in advance. To not forget pauses for questions, it is a good idea to include them explicitly into the informal TRAINING CONCEPT.

To give some relaxing time to the trainer and to involve the participants, one can ask them to contribute: “Do you have an idea for his?“, “Do you know any other example?”, “Does anybody know what I should enter into this field?” Answers can be typed into a chat window.

Most conference systems offer chat windows and trainers are advised to make use of it.

Obstacles
Lose out questions. While the trainer is demonstrating the software tool, participants can ask questions. Since the trainer concentrates on demonstrating the tool, it is easy to miss some of the asked questions. For this reason, the trainer should announce that he will not answer each question immediately but has planned several halting points to discuss questions.

Switching focus To read the questions, the trainer has to switch from the demonstrated software tool to the conferencing system. To avoid this switch, a second CONTROL MONITOR can help. An INVISIBLE CO-HOST could organize and filter the questions.

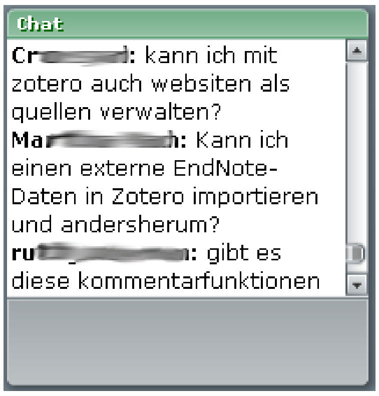
Benefits
- Involvement of participants.
- Questions and obscurities can be addressed.
- Causes social awareness.
- Pauses are relaxing the trainer.

Liabilities
- Questions cannot be answered immediately when they occur.
- Not immediately scanning the questions in the chat window makes it hard to understand afterwards which question refers to which section of the training.
- Pauses can be awkward if no participant asks any questions or gets involved.

Examples
Google Apps training:
User: “Are all users required to have an account at Google?”
User: ”How many people can work simultaneously?”

Flash training:
User: „What kind of braces are required for the command?“
User: “Why is the marker not directly set in the object’s layer? Does that not work?”

LibraryThing training:



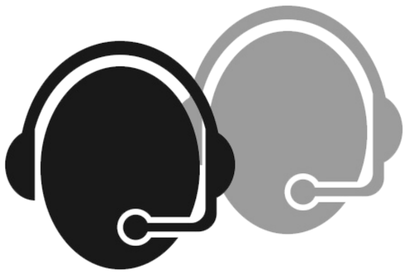
Cr[redacted]: kann ich mit zotero auch websiten als quellen verwalten?
Ma[redacted]h: Kann ich einen externe EndNote-Daten in Zotero importieren und andersherum?
ru[redacted]n: gibt es diese kommentarfunktionen

User: „Can I save website sources in Zotero?“
User: ”Can I import external EndNote-files in Zotero and the other way around?”
User: “Is this commenting feature available for other document types as well?”

Invisible Co-Host

Alias: Assistant

Context
You are running an ONLINE TRAINING with several participants and you can use a chat window for synchronous communication. A chat window allows participants to ask questions and to give feedback whether they understand everything.



Problem
During the training, a trainer concentrates on his screen actions and his audio comments. His
attention cannot be simultaneously and permanently on chat news. Also, chat news can be distracting.

Forces
Awareness Since trainers can not permanently pay attention to the chat window some questions might get lost.

Distraction Participants can chat and communicate with each other. This extends the knowledge exchange and sharing of experiences, however, the trainer is distracted by frequent messages that scroll over the chat window.

Filtering Trainers should address the most important questions at an appropriate time but filtering the most important questions from the chat protocol causes inconvenient delays and binds cognitive resources of the trainer.

Comprehension When checking questions at a later time, for example in a PAUSE FOR QUESTIONS, it is hard to relate general questions to specific training sections, i.e. which demonstrated steps or functions are addressed by the questions.

Solution
Run a training session in a team. Co-located with the trainer is a co-host who acts as an invisible assistant taking care of questions and messages in the chat window. The webcam only captures the trainer, while the co-host works in the back to answer questions, to sort messages and forward the important ones to the trainer.

Details
The training is hosted by a trainer who is an experienced user of a software tool. The assistant does not need to have the same level of expertise as the trainer has. He can answer simple questions directly in the chat window and provide information about organisational issues (e.g. he can answer questions about the duration of the training, whether the training is recorded, how to control the audio volume). His most important task is to filter questions and forward the important ones to the trainer.

Since the co-host always monitors the chat window, he can relate questions in the chat to specific sections of the training (i.e. "this question is about function XYZ").
The co-host sits next to the trainer without being captured by the camera. This way he can act in the background but can directly give signals to the trainer (e.g. wave a note) or write down questions in large letters.

Obstacles
Intrusion The signals of the co-host can distract and irritate the trainer. For mute communication, hand signals can be agreed upon. Thumb up means: "I have seen there is a question and I will address it in a minute." Waving the hand means: "OK, there is a questions, but at the moment I cannot address it because I want to finish the section first."

Briefing The assistant should know the TRAINING OUTLINE. If somebody asks a particular question he can hint that the topic will be demonstrated later.

Benefits
- Relief for the trainer allowing him to concentrate on the training.
- By permanently observing the chat window, questions can be related to specific sections of the training.

- Additional information can be provided in the chat window.
- Concurrent chat makes the training more alive.

Liabilities
- More personal resources are needed.
- Questions and gestures of the co-host can irritate the trainer.
- The co-host needs to have some minimal knowledge about the topic or has to be introduced to it by the trainer.

Examples

Co-Host showing a question

User: “How long will the training last?”
Co-Host: “Approx. 45 minutes”

User: “Is the training recorded? Where can I get the link?”
Co-Host: “Yes, the training is recorded. We will publish the link in our events section at e-teaching.org”

User: “Every now and then the sound is gone”
Co-Host: “You can change the bandwidth to Modem connection. Less quality but continuously audio stream.”

User: “What is the difference between comments and pings?”
Co-Host (using his knowledge): “Comments: Visitors can write text comments. Pings: Other blogs can automatically notify the entry that they are referring it.”

User: “Can I add images and videos?”
Co-Host (using the TRAINING OUTLINE): “That’s coming in a minute…”

Do not disturb!

Alias: On-Air

Context
To run an online training there is no need for a special training room and trainers can host the session from their office.

Problem
Extraneous disturbances such as ringing phones, chatting colleagues, students or any operating noises distract the trainer, interfere the quality of audio broadcast and irritate the training flow.

Forces
Usual Fuss Colleagues, guests, or students cannot know that an online training is broadcasted live from your office and that entering the office or knocking at the door will disturb. In particular if you usually have a policy of open doors and people are used to just entering other people's offices, you have to communicate that there is a "special situation".

Interruption To turn off annoying operating noises or to get rid of someone during the training costs time, destabilizes the Zen of a training session and can cause embarrassing situations in the virtual meeting room.

Absence To leave the desktop temporarily is not an option because a virtual training room without a trainer is very irritating.

Solution
Place a sign at your office door to signal colleagues, guests, visitors or students that you do not want to be disturbed at the moment. Explain the reason.

Details
Block the access to your office by locking the door or place something (e.g. a chair) in front of the door. Tell all your close colleagues that you are running an online training. Turn off phones, mobiles, noisy hard drives or air conditioners.

Obstacles
Explanation To not upset your colleagues, do communicate that your request for silence is not arbitrary but for a good reason. Explain in friendly words why people around you have to be quite for the next 45 minutes.

Defence If an unexpected source of distractions occurs, a CO-HOST can help to turn off a noisy machine, ask people in neighbouring offices to be quiet and get rid of unwelcome visitors. By any means, the trainer should not find himself in a situation where he has to leave his desktop during the training.

Benefits
- Simple and effective, easy to implement
- Everybody knows what is going on
- Running a training without disturbance

Liabilities
- People cannot contact you in an emergency
- Explicit "Do not disturb!" hints can provoke and even invite trouble makers
- People might think you are taking things too seriously

Examples

Radio and TV stations use an „On Air“ signal.

A simple “Do not disturb” print-out decorates the office door when an e-teaching.org event is hosted.

The sign also informs visitors what is going on (e-teaching.org Live-Webcast).

Rehearsal

Alias: Test Run, Dry run

Context
You are planning an online training with several participants and you are currently preparing the structure. The training content and TRAINING OUTLINE are fixed in principle.

Problem
There is a huge difference between planning and running trainings. Only if you run and implement the training, real challenges in operating the software and applying meaningful actions will reveal. Whether the instructional outline works can only be tested if you run the training.

Forces
Complexity You have to try out your PREPARED EXAMPLES to realize that they do not work as planned or are too time consuming and complex.

Elocution The spoken audio comments for each of the process steps have to be clear and to the point. Practice can help. If you are looking for the right words and examples during the training you are risking shakiness in your style of presentation.

Pausing Pauses in online training are much more intensive and seem to be longer; hence, trainers have less time to gather themselves or think about the next steps. Moments of silence (e.g. walking in the training room or looking into the audience) are harder to achieve in virtual environments and often appear to be strange.

Coherence The sequential order of a training requires that information is build on each other and that the structure is coherent. If you only plan theoretically it is easy to lose sight of the big picture and which details have to be presented at which time. As an expert you have everything in your head but that does not mean that you can already communicate it in an instructional way.

Solution
Do at least one test run in advance of the actual training. Use the test run to analyze at which points the training concept can be improved.

Details
It is crucial to speak all comments out aloud in spite of being on your own. By doing so, you get a feeling for the language, how to describe and comment actions. Also, you get a better feeling of how much time you actually need for the training and can adapt the materials accordingly.

To be a skilled user of software is not enough to be a good trainer; you also need to have a good presentation style and skills of communication.

Obstacles
Knowing the traps Known problems of the software to be trained should not surprise the instructor. Rather, he should know the problems and offer solutions. Tackling such obstacles cannot be planned without practically try out the features. The more often you try the same flow of operations, the more likely you will find any potential source of errors.

Passionless It’s the same as with everything: the more practice the better you get. But be aware of running too many test runs. You can easily get bored by the content or start oversimplifying concepts. This could effect your motivation when you are running the real training. You might present the content mechanically. Therefore, the final test run should not be immediately before the real training (e.g. just an hour in advance). It’s better to test run one day in advance.

Benefits
- The trainer gets more confident in demonstrating and explaining.
- Potential sources of errors are recognized and can be addressed.

- The TRAINING OUTLINE is optimized step-by-step.
- If you have a new idea during the test run, you can just write it down and still change the outline.
- You can interrupt a test run at any time.

Liabilities
- Investment of time
- Having too many pre-phrased sentences makes the training monotonously
- Be aware of false safety and be prepared that still a lot of things can go wrong
- Excitement and stage fright will still be with you on the real training

Examples
- Each online training at e-teaching.org is at least tested one time without audience
- Trainings in classroom settings are very often tested with no or just a small group of participants
- Rehearsals are quite common for theatre performances, concerts or public talks.

Training Outline

Alias: storyline, training script

Context
When planning an online training, surely everybody has a mental picture or a concept in his head of what skills should be acquired. Then comes the moment of the real training and you are excited and more stressed.

Problem
While taking the stress – even stage fright – during the live training into account, one easily forgets important information or jumps over some important sections. Thus, some of the knowledge gaps of the participants may remain.

Forces
Skipping It is tempting to leave out some content sections just to get sooner to the end of the training session.

Sequencing Functions and models should build on each other. The sequence in which software features are presented is crucial for its understanding.

Flippancy Flippancy lets one forget to mention some information, leaving behind knowledge gaps of the participants. Delivering information afterwards is laborious and not all participants might receive it.

Biasing Writing a full trainings storyboard costs a lot of time and limits your flexibility as it builds mental barriers to adapt to individual educational situations.

Solution
Make a list of the most important features and process steps you want to communicate. Write them down as an ordered item list setting the structure of the training.

Details
The items will remind you which information to provide in which order. They are the base structure.

Each item is only an anchor for your presentation – and not a pre-phrased manuscript for your training. You should avoid reading information or descriptions from a manuscript.

The item list could include special sections such as defining the learning goals, PAUSES FOR QUESTIONS, or at which time you will show a PREPARED EXAMPLE.

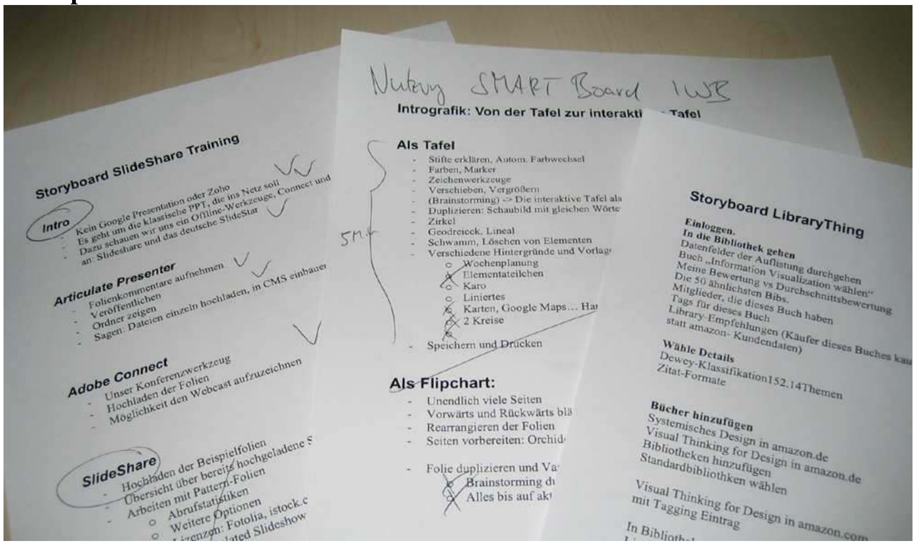
If you run the training with a CO-HOST, you should give him a copy of the training script. If he knows what will be tackled in the training he can better provide information in the chat.

Obstacles
Tracking For your own orientation, highlight the most important information and the start of new sections using a bold font. For clarity, check all items you have talked about with a pen.

Benefits
- Trainings get better structured
- There is less danger to forget relevant information
- PAUSES FOR QUESTIONS can be integrated explicitly
- You always keep on track
- You resist the temptation to jump over some sections

Liabilities
- Blindly following the script makes the training inflexible
- Having too many details and an inner force to necessarily tackle everything can drawn out the training

Examples



Storyboard SlideShare Training

Intro
- Kein Google Presentation oder Zoho
- Es geht um die klassische PPT, die ins Netz soll
- Dazu schauen wir uns ein Offline-Werkzeuge, Connect und
- an: Slideshare und das deutsche SlideStar

Articulate Presenter
- Folienkommentare aufnehmen
- Veröffentlichen
- Ordner zeigen
- Sagen: Dateien einzeln hochladen, in CMS einbauer

Adobe Connect
- Unser Konferenzwerkzeug
- Hochladen der Folien
- Möglichkeit den Webcast aufzuzeichnen

SlideShare
- Hochladen der Beispielfolien
- Übersicht über bereits hochgeladene S
- Arbeiten mit Pattern-Folien
  - Abrufstatistiken
  - Weitere Optionen
  - Lizenzen: Fotolia, istock.c
  - lated Slideshow

Nutzung SMART Board IWB

Intrografik: Von der Tafel zur interakti - Tafel

Als Tafel

5 Min

- Stifte erklären, Autom. Farbwechsel
- Farben, Marker
- Zeichenwerkzeuge
- Verschieben, Vergrößern
- (Brainstorming) -> Die interaktive Tafel als
- Duplizieren: Schaubild mit gleichen Wörte
- Zirkel
- Geodreieck, Lineal
- Schwamm, Löschen von Elementen
- Verschiedene Hintergründe und Vorlag
  - Wochenplanung
  - Elementateilchen
  - Karo
  - Liniertes
  - Karten, Google Maps… Ha
  - 2 Kreise
- Speichern und Drucken

Als Flipchart:
- Unendlich viele Seiten
- Vorwärts und Rückwärts blä
- Rearrangieren der Folien
- Seiten vorbereiten: Orchid
- Folie duplizieren und Va
  - Brainstorming di
  - Alles bis auf akt

Storyboard LibraryThing

Einloggen.
In die Bibliothek gehen
Datenfelder der Auflistung durchgehen
Buch „Information Visualization wählen"
Meine Bewertung vs Durchschnittsbewertung
Die 50 ähnlichsten Bibs.
Mitglieder, die dieses Buch haben
Tags für dieses Buch
Library-Empfehlungen (Käufer dieses Buches kau
statt amazon- Kundendaten)

Wähle Details
Dewey-Klassifikation152.14Themen
Zitat-Formate

Bücher hinzufügen
Systemisches Design in amazon.de
Visual Thinking for Design in amazon.de
Bibliotheken hinzufügen
Standardbibliothken wählen

Visual Thinking for Design in amazon.com
mit Tagging Eintrag

In Biblioth

The outlines of online trainings for SlideShare, SMART Board interactive whiteboards, and LibraryThing.

Prepared Example

Alias: Tested Example, Working Example

Context
You plan and run a classroom or online training. In order to show typical scenarios or actions, you will enter example data for the sake of demonstrating functions or process steps.

Problem
The search for meaningful examples costs time which is not available during the time of training. If you use meaningless (random) data or materials, you can show the functions of a program but you communicate the purpose ineffectively. Careless chosen example data can cause undesired effects or atypical results.

Forces
Limitation You cannot use a huge number of examples in a single training, hence you have to use good examples (in terms of instructional benefit).

Meaning Examples have an important instructional impact and are not just a means to click through program functions. Meaningful examples not only demonstrate the functions but also show their typical uses and benefits.

Time An example can be meaningful but time-consuming, long-winded, and take up too much time of a training session.

Solution
Prepare your examples with care. Use meaningful input data and actions that apply to the typical needs of the trained audience. If needed prepare materials (images, spreadsheet values) in advance, in order to avoid editing times that show no key functions of the software.

Details
The most simple example that does not skip any relevant information might be the best. Try several examples and decide which one fits best.

Never enter meaningless data such as “abc“, “xyz“ or “slkjssw swkjd“ into an input field! Use examples that are appropriate to the semantic of the input field. This is important to associate the right type of input and to demonstrate the purpose and meaning of the input data.

If you show graphic user interfaces that allow direct manipulation, you should not start wildly but create new objects purposefully. Do not just sketch on a graphic panel or create meaningless objects. This will only raise questions such as “Well, fine! But why do I need this?” If the function is meaningful then you will always find a meaningful example!

Obstacles
Stickiness Due to all the efforts for finding good examples one might be tempted to use them at any price. But one should always adapt to the individual training situation. If one example takes longer than planned (e.g. participants asked some questions) just skip another example. If you use a chat window you can ask participants whether there is need for more examples.

Renewal Do not use examples only because you have prepared them! It’s hard to discard an example that has cost you some time to create. But if you find a better one just go for it!

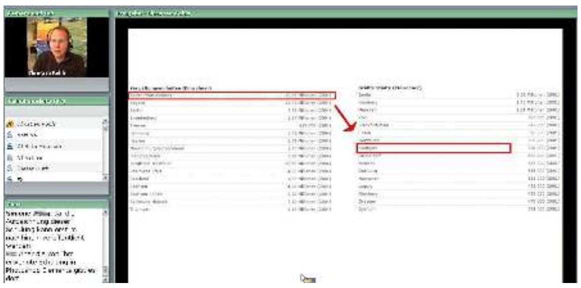
Benefits
- Meaningful examples increase the learning effect
- The trainer is more relaxed during the training if he has not to search for examples
- It helps to avoid unexpected results when using the software
- Good examples can be re-used

- A toolkit of good examples lets you act flexible and offers the opportunity to adapt the training according to the interests of the participants

Liabilities
- It takes a lot of time for preparation
- You may have to adapt examples for each specific group to provide meaningful example data

Examples
LibraryThing training (http://connect.iwm-kmrc.de/p79077399/)
LibraryThing is an online service to manage your library. To demonstrate the service, an account was created and all books of the trainer were added online (in advance) to show a realistic setting. To demonstrate how a book was added, the trainer chose books from which he knew that meta-data were available in other online libraries. In the actual training he only added books to the list which he had successfully added before to avoid any bad surprises. LibraryThing also shows correlations to other users with similar interests (having similar books or tag clouds). To make sure to use interesting examples, the "live" exploration of correlated tags and users was tested several times in advance. The queries that created the best results were used in the actual training.

Animation in PowerPoint training (http://connect.stine.uni-hamburg.de/p65354949/)
PowerPoint offers many ways to animate objects. Usually such animations are distracting but cleverly used they can offer instructional support. This was demonstrated in an online training. Since the training was meant to show how to edit more complex PowerPoint animations, each image and each slide layout was prepared in advance. The purpose of the training was not to show how to create or find interesting images. Therefore, the example image material was prepared in advance.



Animated highlighting of table rows can be used to direct the attention of the audience. In the training it was explained how to create such animated highlighting. The table itself was prepared in advanced because the topic was "how to create animations" and not "how to create tables".

G. Summary (English)

Patterns are an efficient and successful way to capture best practices and access the tacit knowledge of experts. While the basic concept of patterns is simple and applicable in various domains there are many gaps in understanding the deeper meanings of the approach. This work will explore the nature of patterns and pattern languages by discussing the state-of-the-art, linking the pattern concept to its original ideas and develop a theoretical framework to understand the relations and differences between patterns in the world, patterns in our heads and pattern descriptions.

The common ground for patterns is to capture the context, problem, conflicting forces, a generative solution and the consequences in written form. To discuss the meanings of these analytical perspectives a path metaphor will be introduced in this thesis. The concepts will be linked to Alexander's original ideas and common views from the pattern community. We will also identify gaps such as the weak understanding of the different problem levels that should be addressed in a pattern. Moreover, the thesis will reason about the quality of patterns by reflecting on wholeness and its properties.

The major contribution is the foundation of a framework that links the pattern concept to established theories from other disciplines. Patterns in the world are related to the philosophical idea of universals. A thought experiment will show the problems of abstracting and dividing a structure into parts and patterns. The construction of patterns in our heads is explained according to schema theory. Different experiences can lead to different pattern ideas. In our view the descriptions of patterns is similar to the formulation of theories. A pattern is a network of statements about the interaction of forces in a given context. Its proposed solution is supposed to resolve the existing conflict of forces. Such claims are subject to falsification and can be treated as hypotheses about good design. While good patterns are grounded on empirical cases, the generalized forms cannot be relieved from the problem of induction - what has worked in the past does not necessarily work in future cases. Patterns need to be tested continuously and evolve based on new data. From these considerations we will derive practical guidance for pattern mining.

The proposed theoretical framework will be supported by empirical findings. The lessons learnt from an authoring system that generates patterns of interactive graphics show that we can create many variations of instances based on the same universal code. This universal, however, is something artificially constructed by designers. It does not proof that this universal represents the world appropriately. To see whether individuals construct similar patterns we will report on the results from a laboratory study. The experiment tests the construction of schemas of interactive graphics. We will also reflect on pattern descriptions as the result of mining processes. Depending on the format, granularity and level of abstraction these descriptions contain different amounts of testable instructive information. A retrospective document analysis will compare different descriptions of patterns to illustrate this point.

The main contribution of this work is to show the variety of ideas about patterns, the underlying theories and relations between patterns in the world, patterns in our heads and pattern descriptions. These relations show that patterns are socially and mentally constructed. At the one hand patterns try to capture one view of the reality that can be tested empirically. On the other hand pattern descriptions also influence which artefacts are created in the future. There is a feedback loop as patterns capture a perceived reality and potentially shape parts of the future reality, i.e. new objects in the world.

H. Zusammenfassung (Deutsch)

Muster sind ein effizienter und bewährter Ansatz, um erfolgreiche Lösungen und das implizite Expertenwissen über deren Umsetzung zu sammeln und zu dokumentieren. Die grundlegende Idee des Entwurfsmusteransatzes ist leicht verständlich und wird bereits in vielen Fachdisziplinen eingesetzt. Dennoch gibt es häufig viele Lücken, wenn es um ein tiefer gehendes Verständnis des Ansatzes geht. Diese Arbeit wird daher die grundlegenden Eigenschaften von Mustern und Mustersprachen aufzeigen. Dazu werden der aktuelle Kenntnisstand (State-of-the-art) dargestellt, die ursprünglichen Ideen des Ansatzes rückverfolgt und ein Theorierahmen entwickelt, um den Zusammenhang zwischen Mustern in der Welt, Mustern in unseren Köpfen und Musterbeschreibungen zu verstehen.

Der gemeinsame Nenner für alle Entwurfsmuster ist die Erfassung von Kontext, Problemstellung, wechselseitigen Wirkkräften, einer generativen Lösung und deren Konsequenzen in schriftlicher Form. Die Bedeutungen dieser analytischen Perspektiven werden anhand einer Pfad-Metapher erörtert. Die dahinterliegenden Konzepte werden auf Alexanders ursprüngliche Ideen zurückgeführt und mit der allgemeinen Sichtweise der Pattern Community abgeglichen. Dabei werden wir auch Lücken identifizieren, zum Beispiel ein häufig nur sehr oberflächliches Verständnis der verschiedenen Problemebenen, die in einer Musterbeschreibung diskutiert werden sollten. Neben dem Beschreibungsformat werden in dieser Arbeit auch Qualitätskriterien für gute Muster thematisiert, die im Wesentlichen auf Ganzheitlichkeit bzw. Gestalthaftigkeit zurückgeführt werden.

Der wichtigste Beitrag der vorliegenden Arbeit ist die Entwicklung eines Theorierahmens, der den Musteransatz mit etablierten Theorien anderer Disziplinen verknüpft. So lässt sich ein Zusammenhang zwischen Mustern in der Welt und philosophischen Vorstellungen über Universalien herstellen. Mithilfe eines Gedankenexperiments werden die Probleme beim Abstrahieren und Zerlegen einer Weltstruktur in seine Teile illustriert. Das Konstruieren von mentalen Mustern wird anhand der psychologischen Schematheorie erklärt. Daran lässt sich aufzeigen, dass unterschiedliche Erfahrungshintergründe in der Regel zu verschieden konstruierten Mustern und Vorstellungen führen. Wir werden die Position einnehmen, dass das Beschreiben eines Musters viele Ähnlichkeiten mit dem Entwickeln wissenschaftlicher Theorien aufweist. Ein Muster ist nämlich ein Netzwerk von Aussagen über die Interaktion von Wirkkräften in einem bestimmten Kontext. Die in einem Muster vorgeschlagene Lösung soll vorhandene Konflikte zwischen diesen Wirkkräften auflösen. Sowohl die Annahme über das tatsächliche Vorhandensein dieser Wirkkräfte wie auch die Wirksamkeit der vorgeschlagenen Intervention sind falsifizierbar und lassen sich wie Hypothesen über gutes Design behandeln. Gute Muster sind zwar immer abgeleitet aus belegten empirischen Fällen, doch die Verallgemeinerungen führt direkt zum Grundproblem der Induktion: was in der Vergangenheit funktioniert hat, muss nicht automatisch auch in Zukunft funktionieren. Muster müssen daher fortlaufend überprüft und bei neuer Erfahrungslage weiterentwickelt werden. Aus diesen Überlegungen heraus werden wir praktische Empfehlungen für das Aufspüren von Mustern („Pattern Mining“) ableiten.

Der vorgeschlagene Theorierahmen wird durch empirische Arbeiten unterstützt. Aus den Erfahrungen beim Entwickeln eines Autorensystems zum Erstellen interaktiver Grafiken haben wir gelernt, dass es möglich ist, vielfältige Exemplare eines Interaktions-Musters zu erzeugen, die alle auf dem gleichen universellen Maschinencode basieren. Eine solche real existierende Universalie ist jedoch künstlich von Entwicklern konstruiert. Damit ist nicht bewiesen, dass diese Universalie die Welt angemessen repräsentiert. Um festzustellen, ob verschiedene Individuen ähnliche (mentale) Muster konstruieren werden die Ergebnisse eines Laborexperiments diskutiert. Das Experiment prüft die Konstruktion von Schemata über interaktive Grafiken. Zudem wird über die dokumentierten Musterbeschreibungen, die aus „Pattern Mining“-Prozessen hervorgehen, reflektiert. Abhängig vom gewählten Beschreibungsformat, der Granularität und dem Abstraktionsniveau findet man in diesen Beschreibungen unterschiedliche Informationsgehalte und überprüfbare Instruktionen. Eine retrospektive Dokumentanalyse wird sich mit verschiedenen Musterbeschreibungen auseinandersetzen und diese miteinander vergleichen.

Der wesentliche Beitrag dieser Arbeit liegt darin, die Vielfalt an Ideen über Muster, die zugrundeliegenden Theorien sowie die Zusammenhänge zwischen Mustern in der Welt, Mustern in unseren Köpfen und den Musterbeschreibungen aufzuarbeiten und darzustellen. Diese Zusammenhänge zeigen, dass Muster sozial und mental konstruiert werden. Auf der einen Seite stellen Muster den Versuch da, eine bestimmte Sicht auf die Realität einzunehmen, die sich empirisch testen lässt. Auf der anderen Seite beeinflussen Musterbeschreibungen aber auch welche Artefakte in Zukunft erstellt werden. Es gibt also eine Rückkopplung, da jedes Muster eine wahrgenommene Realität beschreibt und gleichzeitig das Potenzial trägt, Teile einer zukünftigen Realität zu formen.