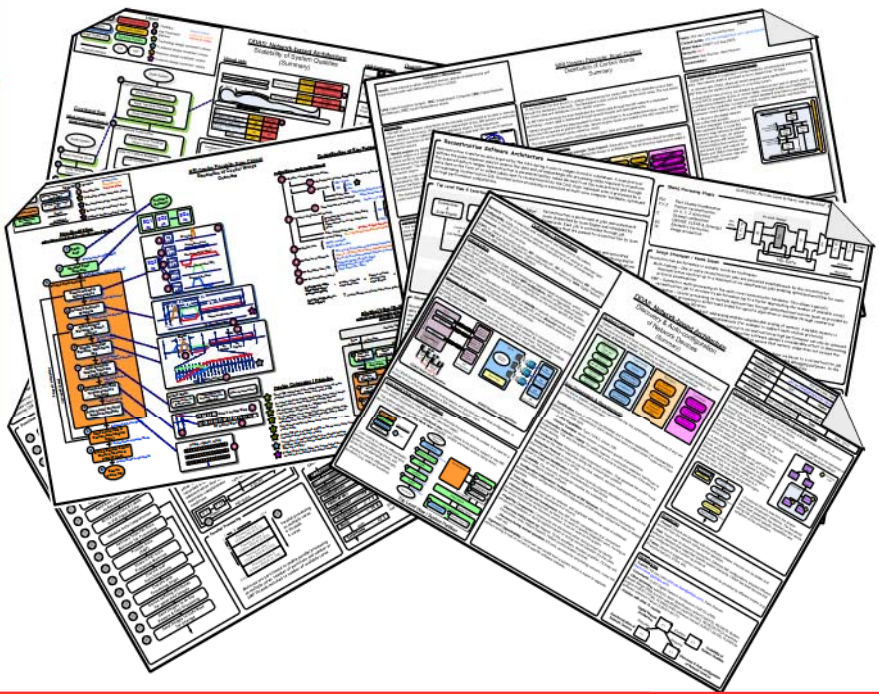
A3 ARCHITECTURE OVERVIEWS

A Tool for Effective Communication in Product Evolution



P. DANIEL BORCHES

A3 ARCHITECTURE OVERVIEWS

A TOOL FOR EFFECTIVE COMMUNICATION IN PRODUCT EVOLUTION

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
Prof.Dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op dinsdag 21 december 2010 om 16.45 uur

door

Pedro Daniel Borches Juzgado
geboren op 11 oktober 1978
te Madrid (Spanje)

Dit proefschrift is goedgekeurd door de promotor Prof.Dr.Ir. F.J.A.M. van Houten en assistent-promotor Dr.Ir. G.M. Bonnema.

A3 Architecture Overviews

A Tool for Effective Communication in Product Evolution

PHD THESIS

By Pedro Daniel Borches Juzgado at the Department of Engineering Technology (CTW) of the University of Twente Enschede, the Netherlands.

Enschede, 21 december 2010

The promotion committee:

Dean Prof.Dr. F. Eising	University of Twente	Chairman, Secretary
Prof.Dr.Ir. F.J.A.M. van Houten	University of Twente	Promotor
Dr.Ir. G.M. Bonnema	University of Twente	Assistant-Promotor
Prof.Dr.Ir. J.I.M. Halman	University of Twente	
Prof.Dr.Ir. G.J.M. Smit	University of Twente	
Prof.Dr. T. Tomiyama	Delft University	
Prof.Dr.Ir. B.R.H.M. Havekort	Universiteit Twente	
	Embedded Systems Institute, Eindhoven	
Prof.Dr. G.J. Muller	Buskerud University College, Noorwegen	
	Embedded Systems Institute, Eindhoven	
Ir. P. van Liere	Philips Healthcare	System Architect

Keywords: Systems architecting, system evolution, architecture, A3, architecture overview, knowledge, effective communication, Philips MRI.

ISBN: 978-90-365-3105-4
Copyright © P. Daniel Borches Juzgado, 2010
Cover design by P. Daniel Borches Juzgado
Printed by Wohrmann Print Service, Enschede
This Thesis has been published using LaTeX 2_ε and the Twentethesis documentclass.
This work has been carried out as a part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

A mis padres, Marian y Daniel, gracias a ellos y su amor incondicional ésta tésis ha sido posible. Ellos son los arquitectos del hombre en el que he evolucionado.
A mi hermano Pablo, compañero de fatigas, y a su mujer, Agata, que siempre han estado ahí para lo que ha hecho falta.
A mi abuela, Mercedes, que siempre ha sido mi mayor fan.

To my parents, Marian and Daniel. Thanks to them and their unconditional love this Thesis has been possible. They are the architects of the man I have evolved into.
To my brother Pablo, my battle companion, and his wife, Agata, as they have always been there when I needed them.
To my grandma, Mercedes, she has always been my number one fan.

Roda de Bará
August 2010

Summary

Creating products from scratch is time consuming and costly. Evolving an existing product is a strategy often chosen by companies to meet customer demands in a timely manner. Evolving a product, however, poses great challenges, especially if the product is a complex system.

The ability of a system to be easily evolved is termed evolvability. Many terms in literature refer to the ability of a system to withstand change. In this Thesis, some of those concepts are reviewed and evolvability is clarified. Evolvability refers to how a system design changes from one generation to the next, such as specifying which aspects of the design are passed down and which are new to the previous generation.

Evolution challenges employees face in an industrial context have been investigated, finding out that managing system complexity, lack of system overview, ineffective knowledge sharing, troubles in finding system information, and difficulties in communicating across disciplines and departments are the main barriers to evolve systems. Those factors, especially the lack of knowledge sharing, have been found to be in many situations the root cause of development problems and poor decisions.

Current strategies to support evolution are based mainly on providing "design rules" to create modular designs, which are argued to be more evolvable, and developing automated ways to estimate the impact of change, so undesired impact of redesigns can be avoided. Other approaches focus on supporting the team in charge of evolving the system by providing them with the right tools. In this Thesis, some of those approaches are tested in real on-going projects at industry. Lessons learned from those experiences will be used to understand what is needed to support evolution of complex systems.

Over the years, technologies, designs and implementations of a system may have changed completely. It may be only the architecture that remains from the original system. Architectures, once consolidated in an architecture description, are of paramount importance for the development and evolution of systems. Architectures provide a framework in which evolution can be performed, enable early analysis of the system, facilitate communication among stakeholders, etc. The concept of architecture and what elements belong to an architecture description are still unclear.

The process of creating architectures is called architecting. This responsibility lies with the architect. We have identified that one of the major needs of architects is to share architecture knowledge. In other fields such as software engineering sharing architecture knowledge has been identified as one of the key factors for project success. Sharing architecture knowledge, however, is not common practice in most companies. Current research to share architecture knowledge is developed from a technology perspective, which is usually not the preferred option of architects. Architects prefer simple, easy to use methods and tools. In addition, most solutions are not tailored to the architecting process at a particular company, or do not fit in an industrial context.

Even if a knowledge sharing mechanism succeeds in delivering architecture knowledge, if this knowledge cannot be effectively communicated to the variety of stakeholders, it can render a great architecting work ineffective. Effective communication in the architecting

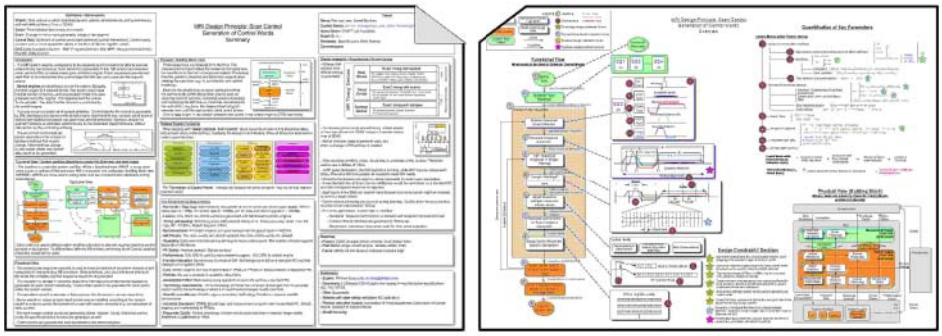


Figure 1: A3 Architecture Overview Example. Left: A3 Summary; Right: A3 Model

context means that individuals and teams understand the essential aspect of the architecture knowledge other individuals or teams want to share.

Communication of architecture knowledge is perturbed by "architecture noise". This noise is caused by human factors such as limited processing capabilities, and organizational factors such as company location. Any mechanism designed to share architecture knowledge should take those factors into account so architecture noise is prevented and knowledge communicated effectively.

The A3[1] Architecture Overview is a tool designed for effective communication of architecture knowledge. In the design of the A3 Architecture Overview, human and organizational factors, as well as experiences in the use of other tools have been taken into account. The aim of an A3 Architecture Overview is not to be complete, formal or executable. An A3 Architecture Overview is meant as an artifact to support effective communication of architecture knowledge. As shown in Figure 1, an A3 Architecture Overview uses two sides of an A3 sheet. One side displays a structured model (A3 Model), composed of several interconnected views, while the other side displays structured textual information (A3 Summary). Visual representations are encouraged to aid understanding. Structure in the A3 improves readability and comprehension.

A Reverse Architecting method is provided to collect, abstract and present the implicit architecture knowledge spread within the company, in the form of A3 Architecture Overviews.

The A3 Architecture Overview tool has been used in real industrial projects at Philips Healthcare MRI department as a communication tool to share architecture knowledge. It has shown to be an improvement to the traditional way in which companies share architecture knowledge. When compared with architecture documents as a means to share architecture knowledge, which is usually the most common form to share knowledge at companies, the A3 Architecture Overview has proven to be more readable, more understandable, have better usability (e.g. at meetings) and have a more adequate amount of information.

Based on observations and feedback from users obtained in different surveys at Philips Healthcare, the A3 Architecture Overview has proven to be useful to mitigate evolution barriers, to meet architect's needs, to fit in an industrial environment, to be easily incorporated into the architecting process, and to provide additional benefits such as capturing architecture insight. Those results show that the A3 Architecture Overview is a powerful tool for effective communication in product evolution.

[1] A3 is an international paper size standard of 297 x 420 mm (American metric equivalent of 11 x 17 inches)

Preface

This is no ordinary Thesis. At least that is what I have been told by those who have read it. This research done in this Thesis is aimed to expand the Systems Engineering body of knowledge. This work in particular focuses on the human side of Systems Engineering, which is often left aside by many other engineering fields. This human side is very hard to measure or quantify, and even harder to validate. For that reason, you will barely find formulas here, nice graphs or simulations. You will find however facts based on experiences in real projects, feedback from real practitioners, and observations of real-life situations in a company. Whether you are an architect, designer, manager, engineer, or just a person who needs an approach to communicate some knowledge, I think you will find in this Thesis a valuable tool that you can use in your daily work.

"Scientific" is usually synonym to complicated and hard to understand. The more complicated the more scientific it looks. Scientific is usually related to formulas, equations, simulations, graphs, and such things. Sometimes this is taken to the extreme; if there is no doubt a research work looks scientific, even when the research value is unclear, it gets accepted. In this Thesis it is just the opposite; people see the value immediately, however as it is easy to read and easy to understand people worried about the scientific value. Scientific validation is always a challenge in this kind of research. The most well-known book in this field is named *"The Art of..."*, highlighting that many of the work done in this field resemble more an art than a science. This is even more challenging when the research is done in an ever-changing industrial environment, where situations and experiments are hard to replicate. This, however, does not mean that scientific work is not possible. What it means is that there is not a nice formula or graph that clearly validates the work done. Research in Systems Engineering relies on observations, experiences, feedback, and similar means for validation. You will find plenty of those in this Thesis.

The most difficult part of this Thesis was making the A3 Architecture Overview simple. It was so easy (and tempting) to add features, functionality, technology (e.g. to develop an automated software tool), and other things that we engineers find so fascinating that I felt into all those pits. After all, by adding those features it felt more scientific. I had to learn the hard way that "new", "software-based" and "different" is not a reason to do things. The fact that the research has been carried out within a company helped me to realize what is needed and what is not, no matter how fascinating or how "publishable" it looks like. If the outcome of the research is not useful or applicable, it is easily put aside.

Although the focus is evolution of complex products and systems architecting, that does not mean this work is only applicable to those fields. The A3 Architecture Overview is tailored to architecting, but it can be easily adopted in other fields. For that it is necessary to find out what is the essential knowledge that is required in a particular field, and define an optimal structure and an effective way to represent the knowledge. You do not have to be an architect or evolving a system to use an A3 Architecture Overview.

Before you start reading this Thesis, a word of advice; **do what I say and not what I do**. In this Thesis I describe a way to support effective communication of knowledge, and to

achieve that I state thinks like that; the amount of information should be limited (to an A3 to be precise), visual attributes should be encouraged, human and organizational factors taken into account, and many other advice based on the work carried out at Philips Healthcare. However, if you read this Thesis, you will find that I do not follow my own advice; this Thesis has too much information, its size is not A3-based, visual attributes are scarce, there is repetition in the text, and much more. The reason (excuse) is that I was constrained by the academic format of a Thesis, and that I was not brave enough to challenge it with an A3 Architecture Overview Thesis (although I almost tried). I hope, however, that you have the bravery to challenge the "traditional" way of consolidating knowledge in your work and try the A3 Architecture Overview, because I do know it works.

Contents

Chapter 1

Introduction

In this chapter we identify the main problem that triggered this research; the evolution of complex products. Terms like evolvability, systems architecting, architecture, knowledge, and communication are introduced. To deal with such a difficult topic, the context in which this research has been carried out; the Darwin project and Philips Healthcare MRI as the industrial partner are presented. Finally, the research itself is introduced; the focus and scope, the goal, the research questions, the research approach, and how to evaluate this research.

1.1 The Problem

In today's competitive environment, companies struggle to reduce time to market for new products and to stay ahead of competition. Companies are pressured to shorten their product development cycles, increase the performance of their products, incorporate new features to cope with increasing customer demands, while reducing their development budgets. Since developing a system from scratch is time consuming and costly, a strategy often chosen is to create new products by evolving an existing one (see Chapter 3). This strategy enables companies to reuse existing infrastructures and available knowledge, while focusing their resources on product improvement.

Nowadays, designing a system that is easily evolved is considered best practice in many industry domains. With evolvable systems, companies can benefit from a system that can adapt to design changes at a cost less than is required to build a new system. The ability of a system to be easily evolved is termed **evolvability**.

Research regarding evolvability of complex systems is scarce. Most research regarding evolvability and evolvable systems focuses in establishing guidelines or "design rules" during the design process so evolvability concepts can be built into the system (e.g. a modular system is expected to have a higher degree of evolvability). Other approaches focus on how to estimate the impact that a design change may have in the system, so undesired consequences can be avoided. Another trend of research focuses on how to support the person responsible of evolving a system; the **system architect**. By helping architects to best use their abilities in their duties, they can make better decisions, enhancing the evolution process.

During the evolution of a product, changes in the design are inevitable. Technology evolves, engineers find better ways of doing things, design problems are identified, new features incorporated, etc. Consequently, products have increased in complexity over the years, as well as the organizations that develop them. Evolving complex systems requires multidisciplinary teams, as no single individual or group can handle or understand the whole system. Complexity leads to non-trivial dependencies across system and organizational boundaries. As a result, the impact of design changes can have extensive consequences due to unknown or hidden dependencies within the system and the organization.

The understanding that a company has about the impact that design changes have on the system determines its ability to cope with system evolution. That understanding enables the company to focus the development effort in adding value to the system rather than dealing

with unexpected problems that may arise in the process. For understanding, knowledge should be available to all team members. Typically some of this knowledge is explicit in the form of text documents; however most of this knowledge is implicit in experts' minds. Companies already have a large amount of knowledge about the domain, yet few companies know how to capture the implicit knowledge effectively, and even fewer companies know how to reuse that knowledge. Decision making therefore fails to take advantage of the knowledge the company already has.

Sharing of knowledge is therefore, essential. By helping the different teams to share and reuse the knowledge the company already has, they are able to make better informed decisions, enhancing the evolution process. Current support to knowledge sharing however is very limited. Consequently, evolvability is usually delegated to the architect's experience and intuition.

Figure 1.1: Car Evolution

Dealing with evolution is complicated, as implementations, technologies, and designs may have changed completely over the years. As apparently everything from the original system has changed, that makes it hard to know which knowledge should be captured to support evolution. Something from the system however has probably remained; the **architecture**. The architecture of a highly evolbable system is likely to not change at all. For example, as shown in Figure 1.1, while the implementation, the technologies, and the design of a car have changed completely over the years, there are some aspects of the design such as the use of wheels to move the car, an engine to power it, the need of room for a passenger, etc, that have remained. Those aspects belong to the architecture and are present in the different car evolutions. While implementations of the system may have changed several times during the lifespan of a system, it may be only the architecture that remains from the original system.

Creating architectures is often called **architecting**. Architecting is an essential step in the design process. The architecting phase involves determining what the system is supposed to do and how specifically it will do it. During this phase vital decisions must be made, and those decisions are the responsibility of the architect. Despite the importance of the architecting phase, there is little support to systems architecting. The architecting process is usually performed by people who have gained experience over the years.

Architectures, once consolidated in architecture descriptions, enable a way to understand the evolution of complex systems, to design them, to manage them, and to provide long-term rationale of decisions made. Architecture descriptions also play an important role in managing communication among stakeholders, as they serve as a reference artifact that can be used to share knowledge about the design and the decisions that led to the system. The knowledge related to architectures is termed architecture knowledge (see Section 4.2.1).

State of the art research about architecting and architectures comprises several activities, such as developing conceptual modeling languages, software tools, applications for storing design decisions, repositories and databases search engines, "intelligent advice", applications to support writing documents, software to perform architectural analysis, and text mining of stored knowledge (see Section 4.1.1). Most research effort focuses on supporting the storage of architectural knowledge, and little on how that knowledge is communicated and consumed by the various stakeholders. A great architecting work can become useless if architecture knowledge is not effectively communicated to the stakeholders.

Research in Systems Engineering has shown that successful companies in product development have enhanced their design and development process by improving communication between and within teams (see Chapter 5). Communication is improved by making available knowledge explicit, in a fashion that enhances visibility on key issues and by having effective design reviews more often. The goal is to promote collaboration across disciplines, to facilitate reuse of knowledge, and to allow teams to become more aware of how each part of the system's design impacts one another. However, how to capture implicit knowledge, and how to effectively communicate it remains a challenge.

There are organizational measures to reduce communication barriers such as allocation of experts to projects, design meetings, project reviews, etc. Those measures however do not eliminate communication barriers and do not create a synergetic way of working. The communication process is an essential part of the design process, as ideas, opinions and views are exchanged to share knowledge, to solve a problem or to design a solution. During the decision making process, architects must communicate and share knowledge with a variety of stakeholders. By understanding the communication process, architects are more likely to achieve their objectives of knowledge sharing, influencing attitudes or to persuade on specific decisions.

The goal of this Thesis is to find a way to support effective communication in order to enable sharing of architectural knowledge to enhance product evolution. By making the implicit architecture knowledge within a company explicit, in a fashion that enables effective communication, we aim to support shared understanding and a way to estimate the impact that a change may have on the system. For that, we have designed a reverse architecting process to consolidate implicit knowledge, and a tool to support effective communication during product evolution; the **A3 Architecture Overview**. The proposed reverse architecting process and the A3 Architecture Overview tool are the result of experiments, experiences, feedback and observations from different projects carried out at Philips Healthcare MRI. The A3 Architecture Overview aims to aid the architect in communicating the architectural knowledge he needs to share to successfully evolve a system. A good shared understanding of architecture knowledge allows the architect to factor into the overall design process the important design issues that can have a big impact on the overall success of the evolved system. In addition, armed with this architecture knowledge, teams can make more informed decisions, better balance competing or overlapping requirements and constraints, and make sure that the evolved system meets customer needs.

1.2 The Darwin Project

The Darwin project had the objective to develop methods and tools for optimizing system evolvability. Although architecture was considered important, the Darwin project did not limit itself to architectures alone. Since the influence of an architect is not limited to the architecture, the Darwin project also focused on what an architect can do to improve evolvability.

To this end, a consortium of industrial and academic partners was set up to carry out the Darwin project with the Embedded Systems Institute (ESI) having the Project Management responsibility. The partners are Philips Healthcare - MRI division (Carrying Industrial Partner), Philips Research, **University of Twente**, Delft University of Technology, Eindhoven University of Technology, University of Groningen (RuG), and the Vrije Universiteit Amsterdam. For the majority of the project time, the researcher was located at the Philips Healthcare facilities in Best.

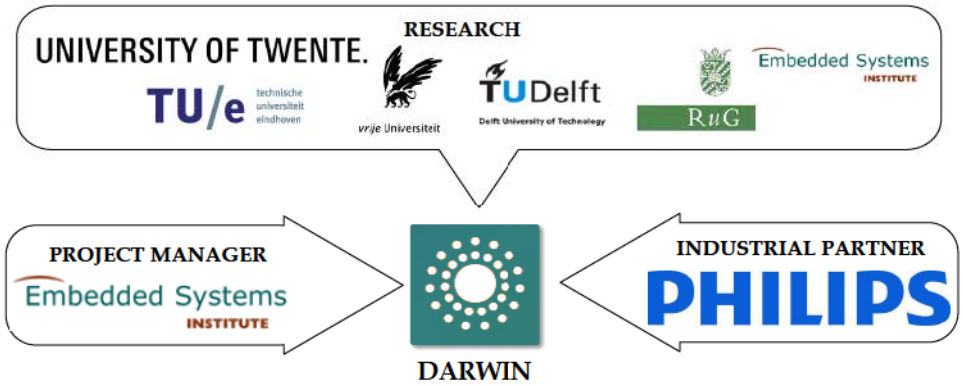


Figure 1.2: Darwin Consortium

The team members had different responsibilities in the Darwin project. The Embedded Systems Institute was responsible to capture and consolidate the knowledge generated in the project. Philips Healthcare MRI was the Carrying Industrial Partner pulling for solutions that fitted its industrial context. Philips Healthcare MRI provided access to technical and business experts and to its repositories containing large amounts of industrial data and knowledge. Furthermore, Philips Healthcare MRI provided feedback on the appropriateness of proposed methods. **University of Twente**, Philips Research and other universities were the solution providers aiming to develop and prove methods that solve industrial problems.

1.2.1 Industrial Partner: Philips Healthcare

Koninklijke Philips Electronics N.V. (Royal Philips Electronics N.V.), usually known as Philips, is one of the largest electronics companies in the world, founded and headquartered in the Netherlands. In 2009, its sales were 23.18 billion Euros. The company employs 123,800 people in more than 60 countries [Philips, 2009]. Philips Healthcare is a global leader in diagnostic imaging systems, healthcare information technology solutions, patient monitoring and cardiac devices.

Philips Healthcare[1] is headquartered at Massachusets (USA) and its development site is located at Best (The Netherlands). Founded in 1896, Philips Healthcare has 32.500 employees worldwide. Philips Healthcare offers diagnostic imaging systems, healthcare information technology solutions, patient monitoring and cardiac devices. Among those products it is the Magnetic Resonance Imaging system (see Chapter 2), which is the case study for this Thesis.

1.3 Focus and Scope of the Research

The scope of this Thesis is on **S**ystems **E**ngineering (SE). Systems Engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed. Systems engineering deals with processes and tools to handle such projects, and it overlaps with both technical and human-centered disciplines such as control engineering and project management. Systems Engineering focuses on concepts like architectures, complexity, processes, and the system boundary is very broad. Systems Engineering seeks to apply those concepts to the process of creating systems.

Focus of this Thesis is Systems Architecting. Specifically on how to support the person in charge of this process; the architect. This Thesis aims to expand Systems Engineering and Systems Architecting body of knowledge by providing means to assist architects during the system evolution process. The research presented in this Thesis does not focus on the system itself (redesigning the system to be more evolvable) but on the architect who has to evolve the system, and how to support his[2] work so the evolution process is enhanced.

1.4 Research Goal

The philosophy behind this Thesis is not to create an automated tool or software application that automatically deals with evolution problems. The goal is not to take away tasks or responsibilities of architects, but to support them during the evolution process.

The goal of the research presented in this Thesis is to design an approach, supported with tools that support architects during the evolution process by:

- providing a way to consolidate knowledge;
- representing architectures for a wide variety of stakeholders;
- communicating effectively with a wide variety of stakeholders, and;
- sharing architectural knowledge in a fashion that triggers discussion and enables collecting feedback.

In summary, we try to find a way to support architects to evolve a system by enabling easy sharing of knowledge and by providing an mechanism to effectively communicate that knowledge to the variety of stakeholders involved in system evolution. By achieving that, we aim to support evolution by making better informed decisions.

1.5 Research Questions

In this Thesis, we try to provide answer to the following **R**esearch **Q**uestions (RQ):

[1] Formerly Philips Medical Systems
[2] he, his, him will be used from now on to reference he/she, his/hers, him/her

1. RQ: What are the main challenges to evolve complex systems? (Chapter 3)
2. RQ: What is needed to support architects to evolve complex systems? (Chapter 4)
3. RQ: Which are the requirements a tool should meet to support effective communication of architecture knowledge? (Chapter 5)
4. RQ: Which are the challenges when communicating architectural knowledge? (Chapter 5)
5. RQ: Are popular approaches to support evolution applicable in an industrial context? (Chapter 6)
6. RQ: Are current ways of capturing architectural knowledge in industry effective? (Chapter 6)
7. RQ: How should knowledge be captured and presented to support effective communication? (Chapter 7)
8. RQ: Can we design a tool that supports effective communication and that is applicable in industry? (Chapter 8)

1.6 *The Research Approach: Industry-as-laboratory*

This Thesis, and the project in which it was developed (see Section 1.2), aimed to develop tools and methods to support evolvability in an industrial context. For that, the research approach *Industry-as-Laboratory* was used. As in other research approaches like *action research* [Baskerville, 1999], used in clinical research, researchers and practitioners work together to find improvements in a real life situation, which is too complex to transfer to a research environment [Avison *et al.*, 1999].

Industry-as-Laboratory is different from other approaches such as traditional research or *case study research* in which the research method depends just on observation, interviews, documents and the researcher's impression [Jaring *et al.*, 2004]. In the *Industry-as-Laboratory* research approach, the researcher works together with an industrial partner in close collaboration with a practitioner (system architect for this research). The practitioner guides, mentors and supports the researcher. In the Darwin project, the researcher is given the role of project leader in real assignments (Advance Development plans), aligned with company interests to solve real problems of the company. In this sense, the researcher becomes part of the company "staff", enabling the author to experience the context and obtain feedback firsthand. In the *Industry-as-Laboratory* approach, validation of the research happens when the outcome of the research is applied in the industrial context, and ideally becomes part of the company "toolkit" [Muller, 2010a].

As summarized in Table 1.1, the *Industry-as-Laboratory* approach presents a series of benefits, as well as some drawbacks both for the researcher and for the industry.

For Researchers — The *Industry-as-Laboratory* approach enables researchers access to industrial knowledge and repositories. It also allows access to state-of-practice approaches and solutions, which provide insight needed to arrive at effective solutions for systems engineering problems. It provides a good environment to collaborate with practitioners. With

Table 1.1: Benefits and drawbacks of Industry-as-Laboratory research approach (based on [Laar, 2010])

For Researchers	
Benefits	*Drawbacks*
- Access to industrial knowledge and repositories - Access to state-of-practice approaches and solutions - Opportunities to observe and to collaborate with practitioners - Insight in industrial problems and their urgency - Early feedback	- Involvement of more stakeholders in the research - Pressure to produce company deliverables - Limited commitment of practitioners in non-urgent matters - Conflicting interest concerning publications (confidentiality)
For Industry	
Benefits	*Drawbacks*
- Focus on the long term - Insight on technology advances - Challenge of assumptions - Unbiased feedback	- Need of specific documents (no-scientific papers) - Outcome may be scientifically relevant yet not useful for the company - Confidentiality issues

this approach insight is gained in industrial problems and their urgency. This approach also enables early feedback; by closely working with practitioners, researchers get feedback which helps to determine whether the goal is realistic, achievable, and desired, and whether the solution fits in an industrial environment.

A challenge of this research approach for the researchers is the involvement of more persons in the research team than just the researcher and the academic supervisor, such as the project manager and the practitioner. This results in more discussions and more requirements to obtain and maintain a common goal. Another challenge is commitment. Practitioners may have more urgent matters to attend to than the research.

For Industry — A benefit of *Industry-as-Laboratory* for industry is focus on the long term. Companies usually focus on the short term, among others, due to market pressure. As a result, no time is taken to reflect on root causes and how to solve them. Researchers on the other hand focus more on root-causes and the elimination thereof. Another benefit is insight in technology advancements. As keeping up-to-date by reading all the literature available is almost impossible. For company employees, cooperation with researchers provides valuable insights in state-of-art technology without substantial investments. Industry benefits also by challenging of assumptions. Over time a company builds up many implicit and company-wide shared assumptions, and employees rarely have time to challenge these assumptions. Finally a benefit for industry is obtaining unbiased feedback. Researchers are less biased by company policies, management pressure, and company culture. This way they can provide feedback without the bias produced by the company, like a consultant would do.

A drawback for industry is the need for specific documents. Researchers are interested in scientific publications, which have different goals, style and use than a company document. In addition, researchers need to publish the outcome of their research, while industry wants to protect the knowledge gained with it.

1.7 Research Evaluation

To evaluate the outcome of this Thesis in the *Industry-as-laboratory* research context poses great scientific challenges. As stated in [Bonnema, 2008], scientific research in a field in which the best known book is named *"The Art of..."* is a challenging undertaking. This is even more challenging when the research is carried out in a real industrial environment. In an industrial context, it is nearly impossible to set up scientifically perfect tests to evaluate

contributions to the systems architecting field. Companies are like living organisms that change with time, as priorities, needs, and even people change. In addition, resources used are expensive. Consequently, recreating experiments or setting up situations is almost impossible. Additionally, evolution may take several years, as in the case of MRI systems, and may require hundreds of man-years. Many factors play a role in the evolution process, and it is almost impossible to separate and analyze all the variables present in the evolution process.

Then, the question that arises is; in this context, how to evaluate whether an approach, tool or method supports the evolution process?

The *Industry-as-laboratory* research approach enables the researcher to test and observe results firsthand. Observations then become an important source to evaluate whether something works in industry or not. Another key source to evaluate the research is feedback from practitioners, users and any other person related to the research.

In this Thesis, feedback has been gathered mainly in two ways; first, through face-to-face interviews and secondly, through surveys. While the data gathered from those sources is presented in Appendix A, the insight obtained from those surveys will be used in different chapters of the Thesis to support the findings of the research.

Final validation, as described in Section 1.6, happens when the architect finds the outcome of the research useful, and ideally this is incorporated into the design and development process.

1.8 Thesis Outline

The Thesis is structured as follows. In Chapter 2, the study case; Philips **M**agnetic **R**esonance **I**maging system (MRI) will be introduced.

In Part I of this Thesis, we explore the relation between evolution and systems architecting. In Chapter 3, current work regarding evolution and system evolvability is reviewed. Evolution barriers are discussed and architectures are introduced as means to understand system evolution. Chapter 4 reviews systems architecting. Duties and needs of architects are discussed. The concept of architecture knowledge is introduced. Chapter 5 deals with effective -human- communication, specifically that of architecture knowledge. In Chapter 6 experiences of applying different approaches to support evolution at Philips Healthcare are presented, and lessons learned from those experiences provided.

In Part II of this Thesis a tool designed to support effective communication of architecture knowledge, the A3 Architecture Overview, and the process to collect, abstract, and present the architecture knowledge spread within a company are introduced. In Chapter 7, a reverse architecting process is introduced in order to recover architecture knowledge and to present it in a fashion that enables effective communication. Chapter 8 describes the tool designed to consolidate architecture knowledge, the A3 Architecture Overview. In Chapter 9, a step-by-step guideline is provided to guide practitioners in the creation of A3 Architecture Overviews.

In Part III of this Thesis, the A3 Architecture Overview is applied in an industrial context and evaluated. Chapter 10 provides the real industrial cases in which the A3 Architecture Overview has been used as a means to support effective communication, and lessons learned from those experiences. In Chapter 11, based on the feedback from practitioners and A3 Architecture Overview users and creators, the A3 Architecture Overview is evaluated as a tool to support product evolution. Chapter 12 concludes the Thesis with conclusions, discussion, recommendations and future work proposals.

Chapter 2

Philips Magnetic Resonance Imaging System (MRI)

In this chapter the Philips Magnetic Resonance Imaging system and the Philips organization are presented. Basic MRI concepts are explained, and the Philips MRI system is introduced as an interesting case study of evolution of complex systems. Some evolution challenges observed at Philips Healthcare MRI are discussed. Finally, this chapter discuss present and future MRI directions and the need for evolvable systems.

Magnetic **R**esonance **I**maging (MRI) has evolved from unpromising beginnings in the 1970s to become nowadays the imaging method of choice for a large proportion of radiological examinations. MRI is an imaging method based principally upon sensitivity to the presence and properties of water, accounting for up to 70% to 90% of most tissues. The properties and amount of water in tissue can alter dramatically with disease and injury. This makes MRI very sensitive as a diagnostic technique. It can be used not just to image anatomy but to investigate organ function, to probe in vivo chemistry and even to visualize the brain thinking [McRobbie *et al.*, 2007].

(a) Philips MRI (b) MRI Images

Figure 2.1: MRI System (Courtesy of Philips Healthcare)

MRI is based on the principles of **N**uclear **M**agnetic **R**esonance (NMR). In 1971, research showed that the magnetic relaxation time of different tissues differs, enabling different tissues to be distinguished. This discovery enabled magnetic resonance (MR) for scanning the inside of the human body. It was during the late 1960s that Philips started to conduct their own MRI research and produced the world's first head images using the MR principle in 1972.

2.1 Principles of Magnetic Resonance Imaging

The principle of Magnetic Resonance Imaging lies in the directional magnetic field, or moment, that is associated with charged particles in motion. Nuclei containing an odd number of protons and/or neutrons have a characteristic motion or precession. Because nuclei are charged particles, this precession produces a small magnetic moment. Hydrogen has a significant magnetic moment and is abundant in the human body. For that reason the hydrogen is the most common element used in clinical imaging.

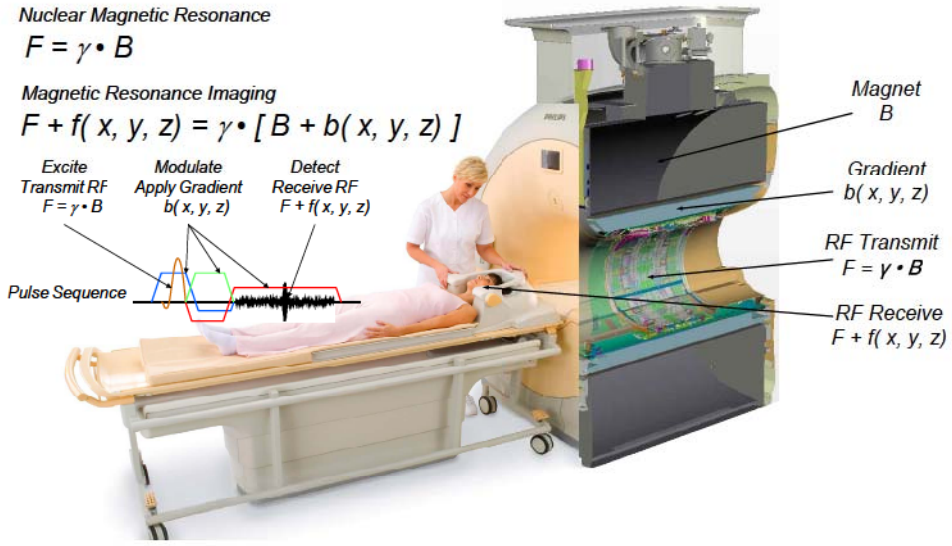


Figure 2.2: Magnetic Resonance Imaging

When a human body (or any other body) is placed in a large magnetic field, many of the free hydrogen nuclei align themselves with the direction of the magnetic field. The nuclei precess about the magnetic field direction like gyroscopes. This behavior is termed Larmor precession. The frequency of Larmor precession is proportional to the applied magnetic field strength as defined by the Larmor frequency (**F**) (see Figure 2.2). While **gamma** is the gyromagnetic ratio and **B** is the strength of the applied magnetic field. The gyromagnetic ratio is a nuclei specific constant.

To obtain an MR image of an object, the object is placed in a uniform magnetic field. Next, a radio-frequency pulse is applied with a frequency equal to the Larmor frequency. Once the RF signal is removed, the nuclei realign themselves such that their net magnetic moment. This return to equilibrium is referred to as relaxation. To produce an image in a specific region, the signal must be encoded for each dimension. The encoding in the desired direction is accomplished by adding a gradient magnetic field (**b(x, y, z)**) to the magnetic field. During relaxation, the nuclei lose energy by emitting their own RF signal at different frequencies (**F + f(x, y, z)**).

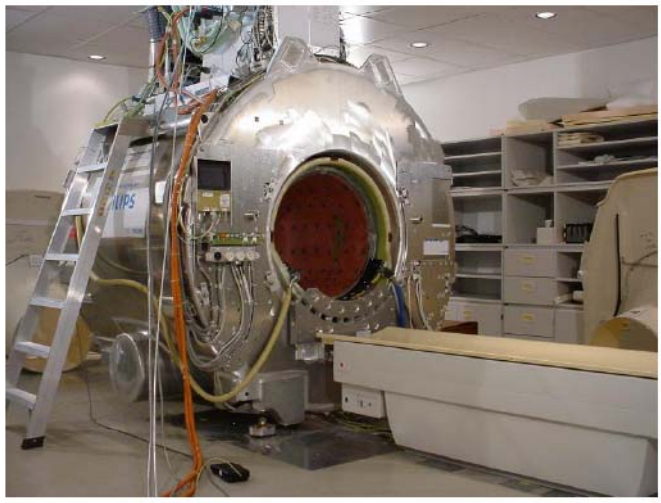
A Fourier Transform is then used to transform the encoded image to the spatial domain. The intensity of a given tissue type (i.e. white matter vs gray matter) depends on the proton density of the tissue; the higher the proton density, the stronger the RF receive signal.

2.2 MRI System

The MRI department of a hospital is arranged differently from the remainder of the imaging department, due to its peculiarities (e.g. vibrations and RF signals may affect image quality). It is likely to have its own dedicated reception, administration, waiting and patient handling areas. The MRI system itself is distributed between three rooms as shown in Figure 2.3; the Examination or RF Room, which houses the magnet, coils and patient handling; the Technical Room, full of supporting equipment such as amplifiers; and the Operator Room which houses the MRI console from which the operator controls the MRI and patient communication.

(a) Operator Room

(b) Technical Room



(c) Examination (RF) Room (MRI without covers)

(d) Coils

Figure 2.3: Philips MRI System (Courtesy of Philips Healthcare)

Figure 2.4 shows an overview of the MRI system (lateral cut). The cylindrical layers of the machine, and the opening, called bore, with the patient and coils are drawn in cross section.

2.2.1 MRI Components

The biggest component of the MRI system is the magnet, which provides a static magnetic field **B** (see Figure 2.3(c)). This static magnetic field suffers from inhomogeneity. As a homogeneous field is required in the scan region (Field of View, see Figure 2.4), shimming is applied. Passive shimming involves placing pieces of iron into the magnet, and is done at install time of the MRI system in a hospital. Dynamic shimming, by controlling currents in electrical shim coils, can be performed on a per-patient or exam case. The magnetic field generated by the magnet outside the bore area is called fringe field, and is limited by magnetic shielding. Three orthogonal linear magnetic field gradients **b(x, y, z)** are used for spatial localization of the MRI signal to select the region to scan. To generate RF pulses at a specific frequency **F** (Larmor frequency), RF transmit coils are used. Depending on the examination to be performed, the control console determines the signals needed to create RF pulses and gradient pulses. To have the desired effect, these pulses need to be amplified considerably, and this is done in the RF amplifier and gradient amplifier respectively. The positioning of the patient in the bore is also done by the control console and based on the exam to be performed (see Figure 2.3(a)). The RF receive coils detect the MRI signal **F + f(x, y, z)**, which is digitized for use in either the image reconstruction or for control purposes, such as calibrations. A physician or radiologist can view the reconstructed image on the workstation, and when the image quality is satisfactory it can be sent to the picture archiving and communication system (PACS) for future use [1].

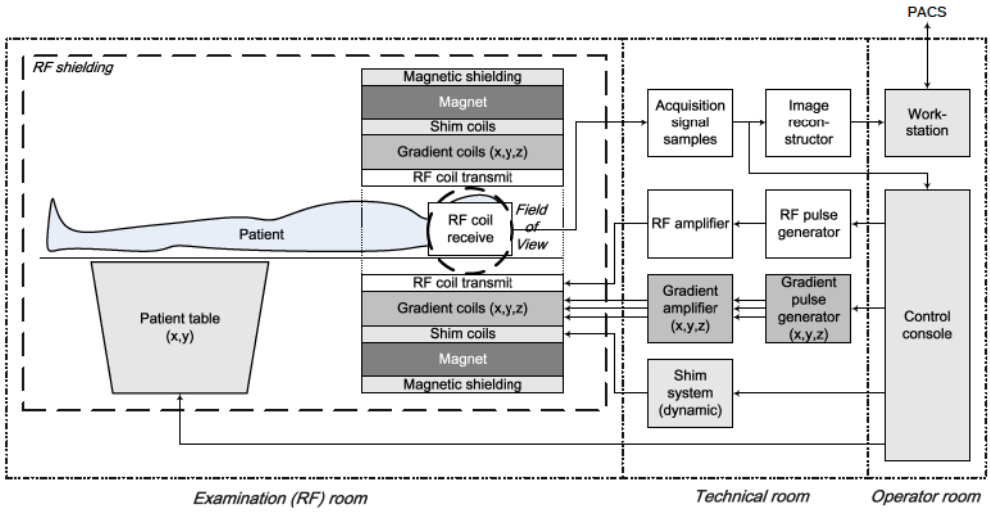


Figure 2.4: MRI Components (Courtesy of Alexander Douglas)

2.2.2 Philips MRI

The Philips MRI system is based on the above components or building blocks, grouped by functionality in so-called chains. A chain is a hierarchically organized, functional unit of the MRI system. It consists of a number of hierarchically lower building blocks and has a unique name and description with a tree structure.

[1] For more detailed information on MRI systems and technology see [McRobbie *et al.*, 2007] [Weishaupt *et al.*, 2006]

The Philips MRI system consists of the following chains:

- **Magnet Chain:** The function of the magnet chain is to produce a strong, static magnetic field. The magnet consists of coiled wires made of super-conductive material, which requires liquid helium to be used as a cryogenic cooling fluid. The imaging capabilities of an MRI system is expressed in terms of its operating magnetic field strength[2]. The strength and homogeneity of the magnetic field created by the magnet influences image quality. Poor homogeneity results in image degradation and artifacts.

- **Radio Frequency (RF) Chain:** The function of the RF chain is to produce the RF pulse sequences (RF Transmit) and to capture the signals produced within the patients tissue (RF Receive). RF pulses are generated by a transmitter coil which surrounds the whole or part of the body. The signals produced in the body as a consequence of those RF pulses are detected using receiver coils. Those signals are very weak and sensitive to electrical interference. Therefore special shielding is built into the examination room (known as Faraday cage) to minimize interferences.

- **Gradient Chain:** The function of the gradient chain is to modulate the magnetic field in the region to scan, by localizing the signals in the body. This is achieved by generating short-term spatial variations in the static magnetic field strength, referred to as gradient fields. The gradient fields are produced by three sets of gradients coils, one for each direction (x,y,z), by applying large electrical currents repeatedly in a controlled pulse sequence.

- **Data and Acquisition System (DAS):** The function of the DAS is to control and to synchronize the MRI chains[3], as well as to generate the data required to generate the pulse sequence. DAS is also in charge of collecting the received signals, process them and transform them into images. Of special interest for this Thesis is the Data and Acquisition system. The architecture and system evolution of this subsystem will be case study of this Thesis (see Figure 2.5).

- **Patient Handling Chain:** The function of the patient handling chain is to position, to monitor, and to observe the patient. Precise patient positioning is achieved by using light alignment markers. For communication, an intercom is provided between the operator's console and the bore of the magnet. Monitoring is performed by physiological measurement equipment, such as peripheral pulse, ECG and respiration, which is used to collect physiological signals that may be used to control the timing of the scan (e.g. to prevent motion caused by breathing).

2.2.3 Philips MRI Knowledge Repository

Philips Healthcare mostly uses traditional documents stored in repositories to capture knowledge. The documentation and archives are structured according to chain hierarchy, resulting in abstract (top-level) archiving and detailed (low-level) archiving. Each chain has one or more building blocks. Top-level archiving is about grouping functionality, whereas low-level focuses on the actual implementation [Jaring *et al.*, 2004].

[2]The unit used is the *Tesla* (T), which equals 10.000 *gauss* (G). The Earth's magnetic field is approximately 0.5G.
[3]Some subsystems require to be synchronized to the nanosecond level.

Building blocks are also used as a management tool to track development. Project deliverables are expressed in building blocks and each building block is subject to design, implementation and test standards. Each building block has an owner assigned who is responsible for the contents of the block. Building blocks that require more than one area of expertise (e.g. software and hardware disciplines), have multiple owners assigned. System architects are responsible for the building block hierarchy.

2.3 *Philips MRI as a Case Study of Evolution of Complex Systems*

The Philips MRI system, as well as the Philips Healthcare organization represents a great case study to learn about evolution of complex systems. In the early days, the MRI was the domain of the physicist and engineers who invented and built it. Because of the diversity of sciences and technologies that gave birth and continue to nurture MRI, it is an extremely difficult subject to master. Authors from the MRI field have stated that a lifetime is not enough to become expert in every aspect [McRobbie *et al.*, 2007]. Within Philips Healthcare MRI, it is estimated that it takes about 5 years to understand the MRI system.

2.3.1 Complex Systems

Complexity is widely studied in many fields of research (see for instance [Axelsson, 2002, Jauregui, 2010, Ottino, 2003]) it is not our goal to repeat those. The adjective "complex" is usually referred to a difficult to understand system, one which behavior is hard to predict, difficult to model, with a large number of parts, etc [Bonnema, 2008]. In this Thesis, a complex system is one that no single person or team can handle or understand completely, and therefore requires a multidisciplinary team or teams for its development and evolution.

2.3.2 Philips MRI as a Complex System

To design and evolve an MRI system requires multidisciplinary teams with competences in areas such as mechanics, electronics, physics, material science, software and clinical science. All the disciplines have to work together on different aspects of the design, such as real-time behavior, control theory, analogue and digital technology, power related issues such as cooling, etc. However, typically people are specialized in a single discipline, and each discipline has specific ways of working, which adds complexity to the design process. The MRI behavior is in itself very complex and involves many parameters and several domains [Weishaupt *et al.*, 2006]. To illustrate the complexity of MRI systems, some indicative numbers of Philips Healthcare MRI products and development are provided in Table 2.1. All this is excluding research, marketing and production floor.

Table 2.1: MRI as a Complex System

MRI Complexity	**Value**
Developers	250 (approx)
Disciplines	Physics, Mechanics, Electronics, Computer Science, Clinical Applications, etc
Development Sites	3
Technologies Used	50 (approx)
Lines of Code (SW)	7MLOC, 10 programming languages (approx)

MLOC = Millions Lines Of Code

2.3.3 PHILIPS MRI EVOLUTION

As shown in Figure 2.5, since Philips released the first commercial scanner back in the 80's, Philips has successfully evolved it several times leading to the present system. The main architecture of the system and the design principles behind it have remained almost unchanged compared to the original system. Implementations and technologies used however have changed completely in those 30 years.

Figure 2.5: MRI Generations and Data and Aquisition System Evolution

If *"the test of a good architecture is that it will last"* [Robert Pinrad, 1993], we can argue that the Philips MRI architecture is a good one. The complexity of the system however, has increased dramatically over the years, making new developments more challenging. As it can be observed in Figure 2.5, time-to-market has increased considerably in the latest generation. It is desirable that evolution problems could be avoided so future systems could be released in the shortest time possible. For that, understanding which the evolutions challenges are and providing a way to cope with them is needed. The need to reuse existing knowledge to provide guidance and prevent problems is more relevant than ever.

Taking all this into account, we believe the MRI system is an ideal case to study system evolution of complex systems.

2.3.4 MRI EVOLUTION CHALLENGES

Manufacturers of medical equipment such as MRI machines have to cope with difficult market conditions. Competition is fierce among manufacturers [MagNet, 2010], and among imaging modalities. The average age of world's population and the life expectancy is increasing, which in general means that more healthcare services should be made available. More functionality and better performance is thus required. For MRI equipment this for example means being able to detect more different substances in the body, a higher imaging resolution, having more examination types available and being interoperable with other imaging modalities. Size and weight of patients are increasing, leading to change in system requirements, requiring a bigger bore[4] and an enhanced patient table. Overall the trend is that the complexity of the imaging modalities is increasing, but due to the larger population requiring healthcare services the costs should be kept low. Besides the functionality increase there are also increasingly stringent demands in areas like safety and reliability. In addition to those challenges, there are also organizational and historical challenges that make evolution difficult.

[4]Power scales to the power of 5 with diameter of bore [Huettel *et al.*, 2004]. To increase the bore with a factor of two while having the same energy reaching the patient (achieving the same image quality), requires a power increase of a factor of 32.

From Incremental Development to Top-down Architecting — From a historical perspective the Philips MRI system has evolved in an incremental fashion. The first experiments with magnetic resonance produced promising results, and the system was extended with more and more components to add new functionality and to solve problems identified. In the beginning this was a logical approach because the system was relatively small. As the system grew over the last two decades, the system became too complex for a few people to oversee, and the system was divided into chains of logically grouped sub-systems like the gradient chain, the magnet chain and the radio frequency (RF) chain.

Steep Learning Curve — There is a relatively long and steep learning curve for newcomers to the MRI field. It takes many years to understand the MRI system. There is not a clear way in place to learn about MRI outside people's field of expertise. Architects are required in many meetings to provide the missing system overview. This illustrates the challenge of transferring the knowledge from experienced architects and developers to new employees for complex systems.

Hard to Estimate the Impact of Change — Because of the logically grouping of the system into sub-systems, teams usually focus on that sub-system only. Yet sub-systems have to fit together to form the total MRI scanner, and this is monitored and guided by the system architects. They have an overview of the entire system and they know parts of the system in detail. The largest part of the development team has very specific knowledge of a small part of a sub-system, but only has limited knowledge of the other parts. This complicates estimating the impact that a local change may have on the system.

Mono disciplinary focus of developers — Even though a developer might be part of a design team working on a certain sub-system, most of the time people develop themselves as a specialist in a specific discipline (e.g. mechanics, software, electronics, thermodynamics). Due to the need of specific competences with deep knowledge of specific areas, people are working on the same field for years, and thus are encouraged to specialize, leaving limited time to learn part of the discipline of other developers.

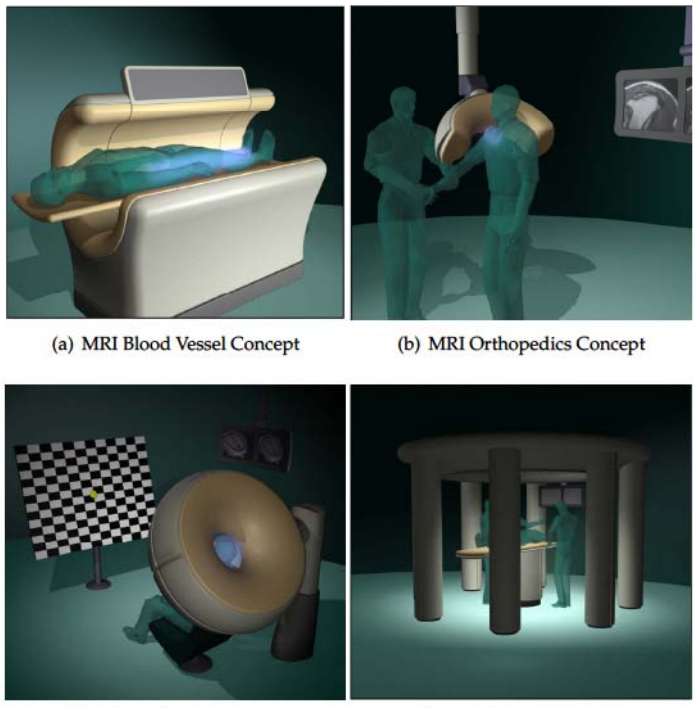
Lack of Overview — Due to the missing of system wide overview and the mono disciplinary focus, it is difficult for most of the developers to estimate what the impact of a change in their part of the system will be on another part of the system. There is thus a gap between the information the system architect needs and the information that the developers can provide. Moreover, the impact of a change brought on by one developer is often hard to estimate. This is also caused by legacy in the architecture. That is, parts of the system that have not been changed for a long time and of which the knowledge in the organization has faded (e.g. experts leaving the company). These parts might be relevant or redundant for today's requirements, but almost nobody can give a quick answer, because nobody is familiar with it anymore, and documentation might be outdated or missing.

Difficulties to Re-partition the System — The previous observations contribute to the difficulties to partitioning the system. The main cause seems to be missing system wide overview for the largest part of the development team (see Section 3.3.2). The person responsible for making the decision on how to distribute the requirements across the sub-systems, the system architect, does not have an in depth knowledge on all sub-systems. Discussing with the

experts can clear up many uncertainties, however due to the mono disciplinary focus of most developers and missing system wide overview of developers, getting reasonable estimates for tasks is almost impossible. Therefore, repartitioning the system is too risky due to unknown impacts that may arise in the process.

2.4 What Next?

Most likely the mail stream will be to continue to build traditional whole body scanners with the main effort being spent on improving quality, adding features, increase bore size, and reduce cost, staying true to the proven architecture. However, with the growing number of examinations in an equally increasing number of applications, dedicated MRI scanners as shown in Figure 2.6 will become available sooner rather than later. Why acquire an expensive full size MRI scanner if there are enough patients to keep a less universal but dedicated scanner busy? This specialization has been observed in other modalities. Similarly, with healthcare budgets being downsized almost everywhere, hospitals will no longer be in a position where they can afford dedicated operators for each modality. This in turn will call for equipment that is "properly trained" in defining its acquisition and geometry parameters all by itself.



(a) MRI Blood Vessel Concept (b) MRI Orthopedics Concept

(c) MRI Psychiatrist Concept (d) MRI Surgery Concept

Figure 2.6: Dedicated MRI Concepts (Courtesy of Philips Healthcare)

In addition, there are many patients who experience anxiety when confronted with the bore. Open systems such as Philips Panorama bring relief for this patient group but there are other solutions as well. For that, among other developments, new system designs are needed. Such designs would have the added benefit of offering excellent patient access for interventional procedures such as surgery (see Figure 2.6(d)).

Whether the Philips MRI system stays with the cylindrical design and/or new designs are created, what it is certain is that the Philips MRI system will evolve. This means that future MRI requirements such as specialization or bigger bore would benefit from adopting approaches to effectively deal with evolution.

I Evolution and Systems Architecting

Chapter 3

Evolution of Complex Systems

In this chapter the concept of evolution and a review of existing work on system evolution and system evolvability are presented. By studying the MRI system and the Philips organization, main development and evolution challenges are identified. Popular approaches proposed to deal with system evolution are introduced. Finally, architectures are proposed as a way to understand system evolution in order to provide the foundations to deal with it, and hopefully incorporate it in the design and development process.

In most industrial sectors, competition is fierce and price erosion is fast. Product manufacturers are under severe pressure to reduce product and development costs and shorten development time, and yet remain technologically ahead of competition. Customer demands for new features to add value to products such as higher performance, improved reliability, more safety and interoperability have been increasing rapidly in recent years. Budgets are reduced despite the increasing need to incorporate increased advanced electronics and software into products. All those demands are leading to an increase in product complexity, which result in additional development challenges. In a study over 140 companies about their experiences in product design and development, it was found that those pressures, as show in Table 3.1, are driving companies to improve their design and development process in order to survive [Boucher and Houlihan, 2008].

Table 3.1: Top pressures driving companies to improve the development process (Source [Boucher and Houlihan, 2008])

Pressures
Shorter product development schedules
Increased customer demand for increased product performance
Reduced development budgets
Accelerated product optimization
Increased requirements to incorporate electronics and software into product

In addition to those pressures, customers demand personalized products. Companies must deal not just with one product but a family of products. This causes an even larger increase in the resources required during the design process, as product families are usually derived from a common platform in order to enable reuse of current knowledge and infrastructures [Muller, 2008].

Since designing and developing complex products from scratch is both time consuming and costly, a development strategy often chosen by companies is to evolve existing products. New systems are built reusing previous system designs or existing platforms. For this reason, the ability of a system to be easily evolved, termed **evolvability** (see Section 3.1.1), is a property vital in the industry for the survival of companies [Isaac and McConaughy, 1994].

Designing a system that is easily evolvable is considered best practice in many industry domains [Schulz *et al.*, 2000]. With evolvable systems, companies can benefit from a system

that can adapt to changing requirements at a cost less than is required to build a new system. Modernization of evolvable systems is expected to take less time and reduce costs. Evolvable systems enable easier insertion of new technology and mitigation of the risk of obsolescence in the system. Life-cycle cost is reduced by a long-lived architecture that eases evolution rather than large-scale system redesign [Laar *et al.*, 2007]. In addition, evolvable products give additional flexibility, as the company can either reuse existing infrastructure to tackle changing requirements or to develop a new product.

Some industrial sectors such as aerospace, automotive, naval and military, have already identified the necessity of adopting evolvability concepts in their development process and have placed it as a primary requirement [Steiner, 1998] [Schulz and Fricke, 1999]. Despite the need for evolvable systems, it is still a relatively unknown field. Nowadays there is no clear way to design evolvable systems and what is needed in the design and development process to enhance evolvability. Evolvability is mainly addressed by increasing system modularity (see Section 3.2.1), and required tasks in design and development process are delegated to the architect's intuition and experience (see Section 4.1.2).

Evolving complex systems however is far from being a simple task. Systems have increased in complexity over the years, as well as the organizations that develop them [Bonnema and Borches, 2008]. This causes non-trivial dependencies across system and organizational boundaries. As a result, changes may have extensive consequences due to dependencies, unknown or hidden within the system and the organization. Even minor top-level functional changes can have lengthy, costly and difficult to predict development cycles. The understanding that a company has about the impact change has on the system determines its ability to cope with system evolution. During evolution, technologies, implementation and the design may have changed. How to deal with a system when apparently everything has changed? The answer lies in the architecture. While it may seem that everything has changed, it may be only the architecture that remains from the previous system.

In this chapter we review existing work on system evolution and system evolvability, aiming to clarify the apparent confusion about what system evolvability is[1]. We investigate the challenges of evolving a complex system by studying the MRI system and the Philips organization[2] (see Chapter 2). We also review popular approaches proposed to deal with evolvability. Finally, evolvability being such a difficult property to deal with, we propose a way to understand it in order to provide the foundations to deal with it, and hopefully incorporate it in the design and development process.

3.1 Evolution and Evolvability in Literature

Evolution as a concept has its deepest roots in biological and social sciences. Most literature regarding the ability to evolve refers to those sciences. Darwin's theory, characterized by heritable variation and natural selection, is often used as a starting point [Darwin, 1859]. For example, in biology evolvability has been variously defined such as *"the ability of a population to produce variants fitter than any yet existing"* [Altenberg, 1994] or *"the genome's ability to produce adaptive variants when acted on by the genetic system"* [Wagner and Altenberg, 1996]. Not surprisingly most of those biological and social theories and approaches are not appropriate

[1] This work is based on the paper presented by the author in the proceedings of *Tools and Methods of Competitive Engineering*, 2008. See [Borches and Bonnema, 2008b]

[2] These findings are published in paper written by the author in the proceedings of *8th Annual Conference on Systems Engineering Research*, 2010. See [Borches and Bonnema, 2010b]

for complex man-made systems as species and systems are not similar enough (e.g. there is no clear analogue of "gene")[3].

In fields close to Systems Engineering such as computer science, since [Lehman and Belady, 1976] laid the groundwork for research into software evolution, this has been an active field of research [Mens *et al.*, 2005]. Evolvability in those fields is defined for instance as *"the capability of software products to be evolved to continue to serve its customer in a cost effective way"* [Cook *et al.*, 2000]. Much can be learned from experiences in software evolution. Yet software and Systems Engineering have differences in goal and scope (e.g. designing complex systems requires multi-disciplinary teams, while designing software may require limited collaboration from other disciplines). Systems Engineering requires a dedicated approach in order to incorporate evolvability concepts into the creation of systems.

In this section we review existing work regarding evolvability. Popular definitions are revised, and other system properties that also deal with change are reviewed. A suitable definition for evolvability is proposed, and existing work regarding how to assess evolvability is presented.

3.1.1 System Evolution and Evolvability

Many years have passed since the first paper regarding "system evolvability" was published [Isaac and McConaughy, 1994][4]. Since then, some theoretical work to understand how systems evolve has been done in the Systems Engineering field [Rowe and Leaney, 1997] [Christian III, 2004]. A few papers have attempted to measure evolvability of complex systems [Christian III, 2004] [Christian III and Olds, 2005], and few more tried to analyze evolvability on computer systems [Rowe and Leaney, 1998] and aerospace systems [Christian III and Olds, 2005]. The importance of adopting evolvability in complex systems has been discussed by several authors [Isaac and McConaughy, 1994] [Rowe and Leaney, 1997] [Ring and Fricke, 1998] [Steiner, 1998] [Christian III and Olds, 2005]. Finally the role that evolvability plays in the system architecture has been described in [Isaac and McConaughy, 1994] [Steiner, 1998].

Despite those contributions, system evolvability is almost an unexplored field. The term system evolvability is used in manifold papers, yet it is used differently depending on the context. System evolvability definition is still open for discussion. In addition, there is no formal way to assess or measure evolvability and evolvability is confused with other system's properties such as changeability [Fricke and Schulz, 2005]. From literature we can conclude that there is almost no real work done in the systems field, and there is almost no research regarding how to support the evolution process and the persons in charge of it.

System Evolvability Definition

Over the years, there have been a few attempts to define system evolvability. Regardless of those attempts, there is still no broadly accepted definition of system evolvability and each author tailors the definition to the specific research carried. Some of those definitions are:

- *"System evolvability is a trait of a system that allows the system to be easily modified due to changes in the environment"* [Percivall, 1994].
- *"System evolvability is a system's ability to withstand changes in its requirements, environment and implementation technologies"* [Rowe and Leaney, 1997].

[3] It should however be mentioned the work of [Dawkins, 1981] who propose analogies for gene outside the biology field

[4] To the author's knowledge. The relation between evolution and complex systems however was introduced by [Simon, 1962]

- *"System evolvability; 'an attribute that bears on the ability of a system to accommodate change in its requirements throughout the system's lifespan with the least possible cost while maintaining architectural integrity"* [Rowe and Leaney, 1998].

- *"System evolvability; the capacity of a system to successfully adapt to changing requirements [IEEE, 1990] throughout its life-cycle without compromising architectural integrity. Furthermore, an evolvable system must meet the new needs of the customer in a more cost effective manner than developing a new system"* [Christian III, 2004].

Each of those definitions tries to identify the main properties that evolvability should incorporate into the system, such as easy adaptation to changing requirements and environments. However they do not agree upon those properties and how they should be addressed. Other definitions of system evolvability are similar to those presented.

In addition to existing evolvability definitions, in literature many terms refer to a system's ability to accommodate change. It is worth to briefly review some of those terms to better understand similitude and differences with evolvability:

- **Adaptability**: *the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.* [Rowe and Leaney, 1998]

- **Flexibility**: *the property of a system to be changed easily and without undesired effects* [Schulz and Fricke, 1999].

- **Changeability**: *the ability to meet changing situations and diversified operations with minimum disruption or delay* [McCay, 1996].

- **Extensibility**: *the capability of being extended resulting in easier, faster, and less costly upgrade in capability* [Bensley *et al.*, 1995].

- **Enhanceability**: *the ease with which new functionality can be added to a system* [Dasgupta, 1991].

More system properties deal with change [Muller, 2004], such as; **portability**, *being able to change the underlying platform*; **upgradeability**, *the capacity of upgrading an entire part of the system with improved features*; **extendability**, *the capacity to add options or new features*; and **maintainability**, *the capacity of maintaining the well-being of the system.* Though this list is not exhaustive it is significant enough to demonstrate that many different terms deal with different, but closely related aspects of change.

However, even if a system can accommodate change one way or another, it does not automatically mean that it is evolvable. While other system properties focus on how to design a system so it can accommodate change during the life-time of the product, **Evolvability refers to how the system design changes from one generation of a product to the next, such as specifying which aspects of the design are passed down and which aspects of the design are new to previous generations.**

Additionally, previous definitions focus only on the system itself, neglecting the context and the design process. Research has shown that systems can be successfully re-architected to incorporate system 'ilities' by considering methods that extend beyond the domain of physical design to include organizations [Richards *et al.*, 2007]. Thus, not only delivered systems or system designs have to incorporate changes to include evolvability concepts,

companies themselves need to incorporate changes within their design and development process. A success in the transition from a non-evolvable system to an evolvable system even without major changes in the system's design might be possible by setting up the appropriate mechanism into the design process. Based on those ideas, within the context of this Thesis, we define evolvability as:

The ability of a system and its context to provide the means to easily identify what aspects of the design can be passed on to a new generation to meet new requirements, with a controlled impact of change.

Taking that definition of evolvability, an evolvable system would be; a system in which designers can easily identify what aspects of the design can be reused to develop a new generation, and what new aspects need to be incorporated to the system to meet new requirements. Also in an evolvable system, the impact of required changes can be estimated so the best design and development strategies can be put in place and undesired impacts can be avoided.

Assessing System Evolvability

Over the years there have been a few attempts to assess system evolvability. Approaches developed to assess system evolvability are based primarily on the work of Mario Bunge [Bunge, 1977]. Although philosophical in nature, Bunge's work has lead to extensive research in ontology's application to the engineering and computer science disciplines.

Regarding system evolvability, [Rowe and Leaney, 1997] proposed a model of systems architecture evolution based on Bunge's ontology. In that model, evolution is considered a type of change and modeled as an 'event'. An event is a pair of states, where each state (start and end) exists in the 'possible state space'. This concept of modeling evolvability implies that both the initial and final evolved states must be known. [Christian III, 2004] tried to use the same approach to measure system evolvability on aerospace systems in order to find out which aerospace designs were more evolvable, by adding some metrics such as figures of merit. However, in a later publication the approach was replaced with another approach based on experts' estimations [Christian III and Olds, 2005]. The method relies on a self-made scale of system evolvability to provide numbers and the experts' understanding of evolvability to select numbers. Despite those contributions, it is not clear why one system is more evolvable than another, and how to use those numbers in future designs. Since then there have been no more attempts to assess system evolvability.

Although in the software community we can find many approaches to measure evolution [Allen and John, 2005], in the Systems Engineering field finding ways to measure evolvability of complex systems is still far from a trivial task. Although finding scales of measure for evolvability is beyond the scope of this Thesis, we believe that any metrics meant to assess evolvability should not only assess system attributes, but should assess the organization as well. As stated in Section 3.1.1, companies play an important role regarding system evolvability; evolvability may be incorporated into the system if a company is equipped with the right mechanisms to support evolution.

3.2 *Supporting Strategies to System Evolution*

Research on system evolvability comprises two main approaches; to drive a system's design in such a way that the impact caused by changes is kept small, and to provide means to improve

the design process in a way that previous knowledge can be reused in future developments. In this section we present research approaches that deal with the first approach, while in Chapter 4 we will dive into the second approach to support system evolution.

3.2.1 Design Rules: Modularity

A field of research that supports evolution focuses on the discovery of heuristics and best practices, also termed "design rules" [Baldwin and Clark, 2000]. These design rules aim to guide designers in their work, so properties such as flexibility or evolvability can be incorporated in the system design [Ernst and Armin, 2005]. According to those design rules, evolvable systems are achieved by adding higher degrees of modularity. It is to say, a system with a modular design enables evolvable systems [MacCormack *et al.*, 2008].

The link between modularity and evolution was first stated in [Simon, 1962]. In Systems Engineering, modular design is an approach that subdivides a system into smaller parts (modules) that can be independently created and then used in different systems to drive multiple functionalities. Modularity is often referred to as the degree to which system components may be separated and recombined, and the degree to which the system architecture enables (or prohibits) those combinations [Schilling, 2000].

The challenge of modularity is how to identify the modules. According to authors such as [Isaac and McConaughy, 1994], evolvable systems should be built on the aspects of the system which are likely to remain unchanged. Those *"islands of architectural stability"*, once in the form of modules, should enable a complex system to evolve quicker [Percivall, 1994]. By designing systems with independent modules, each of them can evolve at a different pace without affecting the other modules [Baldwin and Clark, 2000]. Modular designs are supposedly loose-coupled, so changes can be made in one module without impacting the others, preventing change to ripple through the system. In literature, popular examples of modular systems are cars and computers.

Although there are documented experiences in fields such as software engineering of the success of this approach [MacCormack *et al.*, 2008], in the systems engineering field the challenge remains on how to identify, design and build modules, especially when system aspects span different domains. This approach is even more challenging when evolving an existing system, in which modularity was not included in the original design.

3.2.2 Estimating the Impact of Change

The task of accurately estimating impact of change is one of the larger engineering problems [Ring and Fricke, 1998], and was also observed at Philips Healthcare MRI (see Section 2.3.4). A field of research focuses on estimating the impact of change and change propagation [Clarkson *et al.*, 2004] [Giffin *et al.*, 2007]. By understanding and assessing the consequences of those changes, design effort can be directed towards avoiding undesired impacts and guide the design of the system to one that enables easy evolution. A few approaches can be found in literature, most of them based on **D**esign **S**tructure **M**atrices (DSM).

The Design Structure Matrix, as shown in Figure 3.1, is a popular representation within the design and engineering communities. DSM is used in a variety of contexts, including product development, project planning, project management, systems engineering, and organization design [Browning, 2001]. A DSM can represent a large number of system elements and their relationships in a compact way that highlights important patterns such as feedback loops, modules and so on.

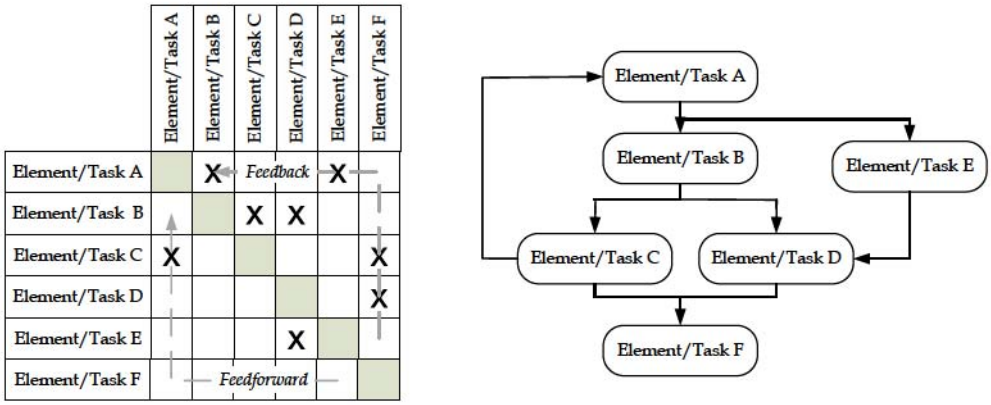


Figure 3.1: Design Structure Matrix Representation vs Graphical Representation

A DSM is useful to capture information to make systematic analysis. In the work of [Clarkson *et al.*, 2004], mathematical models to predict the risk of change propagation in terms of likelihood and impact of change are developed and presented in the DSM cells (see Figure 3.1). With knowledge of likely change propagation paths and their impact on the delivery of the product, design effort can be directed towards avoiding change to "expensive" subsystems and allowing change where it is easier to implement.

In literature, this approach has been tested in real complex systems such as helicopters [Clarkson *et al.*, 2004], in which a 19x19 DSM is created to model the helicopter. However, as it will be show in Section 6.1.3, this approach has clear limitations that make it hard to apply it in an industrial context; the amount of information required is large, and it needs a complete set of input data to produce reliable results. In addition, although suitable for analysis, information within the DSM is very hard to visualize for a human (see Section 6.1.3). Those aspects may limit the utility of this approach, especially in large complex systems.

3.2.3 TRIZ, Trends of System Evolution

TRIZ is an algorithmic approach to problem solving [Altshuller, 1998]. According to TRIZ, every system evolves in the direction of the "ideal system". That is, technical systems evolve over time to higher states of ideality. In the TRIZ framework, patterns have been discovered of how systems evolve, and they are referred to as laws or trends of evolution. Those laws denote general conditions for the creation and development of technical systems and what particular phases of evolution a system passes through. Although the application of a particular trend does not guarantee an increase in ideality of the system, they are meant to predict how systems will evolve into the future.

The main purpose of using trends is to predict the future technology and future evolution of a product or system. Trends of evolution aim to help improving the product in the right direction, and make the product more successful. It is however not easy to use those trends in the development process, especially in complex systems. E.g. *Transition from rigid to flexible to wave technologies.* According to this trend, products are moving from rigidity to flexibility. This means that future stages of a product will be more and more flexible.

3.2.4 Other Approaches

There are some other approaches that can be applied to deal with system evolution,. To name a few:

- Knowledge-Based Design: In this approach, knowledge engineers elicit knowledge from domain experts and build a knowledge-based design system. The system generates designs or supports their creation. In this approach, the amount of knowledge the system possesses and the way it applies the knowledge directly influence the performance of its designs [Liu *et al.*, 1995].

- Lean Development: This approach consist on the application of the principles of the Toyota Product Development System to software or systems development. In theory when correctly applied, lean development results in high quality systems that are developed quickly and at the lowest possible cost [Poppendieck and Poppendieck, 2003].

3.3 *Evolvability in the Design Process*

The design and development of systems has reached a new level of complexity as companies strive to integrate mechanical, electrical and software components (see Table 3.1). Each of those design elements requires specific engineering disciplines with unique knowledge bases, processes and tools. Bringing them all together into a single product or product family is far from a simple task. Coordinating the diverse disciplines required to design a system presents demanding challenges for companies to overcome.

While there is no simple approach to Systems Engineering, there are steps that companies can take to improve the development process with the resources they have at hand. In order to enhance the design process to support the evolution of complex systems, it is necessary to understand what the challenges the company faces during product development are. Then, more specifically, what barriers employees face during the evolution process. Once those barriers are identified, appropriate mechanisms to overcome those barriers can be designed and incorporated into the design process.

In this section we aim to identify main development and evolution barriers, by reviewing existing work done in the field, and by performing our own survey to Philips Healthcare MRI employees.

3.3.1 Product Development Barriers

In the development of complex products, it is necessary to bring together multiple complicated disciplines with little understanding or visibility in their companion areas. Not surprisingly, as shown in Table 3.2, in a study done over 140 companies, companies indicated that the top challenge when developing complex systems is the lack of cross-functional knowledge and qualified systems engineers. This problem arises when a solution for design conflicts is needed, especially when they cross different disciplines. Related is then the inability to understand the impact a design change will have across disciplines, which is another major development challenge also observed at Philips Healthcare MRI (see Section 2.3.4). This is often an expert's limited understanding of the other disciplines involved in the design as well as the lack of an integrated solution.

Table 3.2: Top six challenges of complex systems development (source [Boucher and Houlihan, 2008])

Top Development Challenges
Lack of cross-functional knowledge and difficulty finding experienced system engineers
Inability to understand the impact a design change will have across disciplines
Early identification of system level problems
Difficulty predicting/modeling system behavior until a physical prototype exists
Ensuring all design requirements are met in the final system
Difficulty implementing an integrated solution for all disciplines involved in product development

As stated in Section 3.1.1, a success in the transition from a non-evolvable system to an evolvable system might be possible by setting up the appropriate approach into the design process. In [Boucher and Houlihan, 2008], it was stated that successful companies are less likely to report a lack of cross-functional knowledge as an obstacle. Those companies were able to manage the challenges of systems engineering by providing a multidisciplinary approach and improving communication and collaboration across disciplines. How to specifically address those activities remains a challenge for most companies.

In the study, it is stated that successful companies have adapted their product development in ways to overcome communication barriers and facilitate collaboration. This is achieved by having formal reviews more often with all the disciplines involved in the design, encouraging formal documentation of issues, and developing collaboration tools. The study also states that successful companies are more likely to look at the system as a whole, and more likely to analyze system behavior to determine function or architecture tradeoffs. Unsuccessful companies on the other hand, spend more resources analyzing the impact of changes (see Sections 2.3.4, 3.2.2), however they lack a process to communicate changes as well as collaboration tools to make sure the changes are effectively communicated.

3.3.2 Product Evolution Barriers

In Section 2.3.4, observed evolution challenges from an outsider point of view were discussed. In order to understand why companies face the development challenges mentioned above, it is also necessary to understand barriers employees face when evolving systems. As we discussed in Section 2.3.3, Philips Healthcare has designed several generations of MRI systems for over 30 years. All of those designs are based on the previous system (see Figure 2.5). This means that employees are familiar with the evolution process. To get insight in evolution barriers, a survey to the MRI development organization was performed (see Appendix A).

Table 3.3: Profile of Philips' employees that contributed to the survey

Job Title		Experience	
Managers/Leaders	8	<5 Years	4
Architects	5	5 <Years< 10	13
Engineers	10	10 <Years< 20	9
Designers	7	Since MR Proton	
Domain Experts	2	(>20 Years)	9
Other	3		
Total	35	**Total**	35

The target of the survey was the MRI development organization (250 employees approximately). Marketing, sales, logistics and related departments were not addressed. The goal of

the survey was to identify major barriers employees face when working on the next product generation. In this survey 35 people[5] (around 1/7 of the MRI development population) filled in a questionnaire with 40 questions (see Appendix A.1). To uncover whether background or experience of employees play a role when addressing evolution barriers, those factors, as shown in Table 3.3 were taken into account in the later analysis. From this survey it was found that major evolution barriers employees face when evolving a complex system such as an MRI are, as summarized in Table 3.4:

Table 3.4: System Evolution Barriers (details of the survey provided in Appendix A.1)

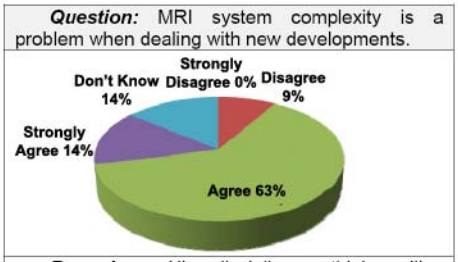
Evolution Barriers	Response
Managing system complexity	Figure 3.2(a)
Lack of system overview	Figure 3.2(b)
Ineffective knowledge sharing	Figure 3.2(c)
Finding the required system information	Figure 3.2(d)
Communicating across disciplines and departments	Figure 3.2(e)

Managing System Complexity — As shown in Figure 3.2(a), most employees found system complexity a problem when dealing with new developments. Many domains are involved in the development process, and the consequences of local changes can impact the system at different levels and different domains. In addition, it was pointed out that legacy implementations increase system complexity. This complexity leads to unclear interfaces between system elements, requirements not clearly transformed into design parameters, lack of budgets, etc.

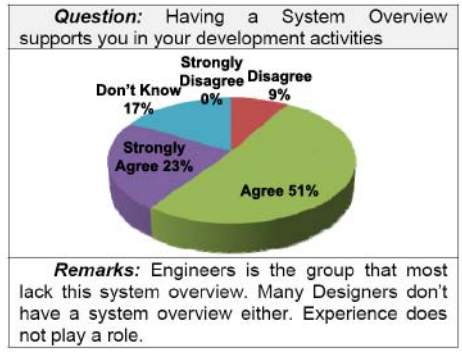
Lack of System Overview — As shown in Figure 3.2(b), and observed already in Section 2.3.4, most employees from all disciplines, regardless the experience they have, felt the need for a system overview to support their development activities. Despite this need for a system overview, it was found that more than half of the MRI development population did not have any kind of system overview. While architects and domain experts claimed they have a system overview, most engineers, designers and managers do not have any. It was also found that, most employees who claimed to have a system overview, this overview was not explicit but in their minds (a mental system overview). Some employees claimed that they acquired a system overview by using old paper diagrams obtained during oral communication.

Ineffective Knowledge Sharing — The essential knowledge required to understand the MRI techniques and designs are usually referred to as design principles within Philips Healthcare. It was found that only half of the surveyed were familiar with most design principles[6]. Among those, it was found that while all architects were familiar with them, other employees such as designers and managers were not familiar with them at all. It was clear from the survey that knowledge about design principles was obtained by experience. That is, the more experience the more familiar people are with design principles. The reason for this may be that existing ways to share this kind of knowledge by using tools such as Wikis are not effective, and therefore this knowledge must be acquired by experience. As shown in Figure 3.2(c), it was found that that this lack of knowledge sharing of design principles was cause of development problems and poor decisions.

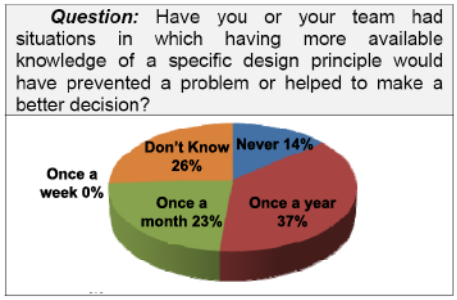
[5]The survey was sent to the whole Philips MRI development population
[6]Described in the SDS, see Section 2.2.3



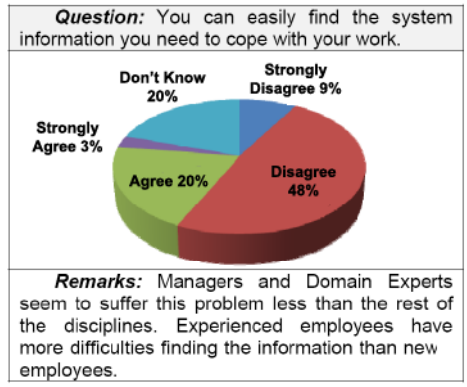
(a) System Complexity



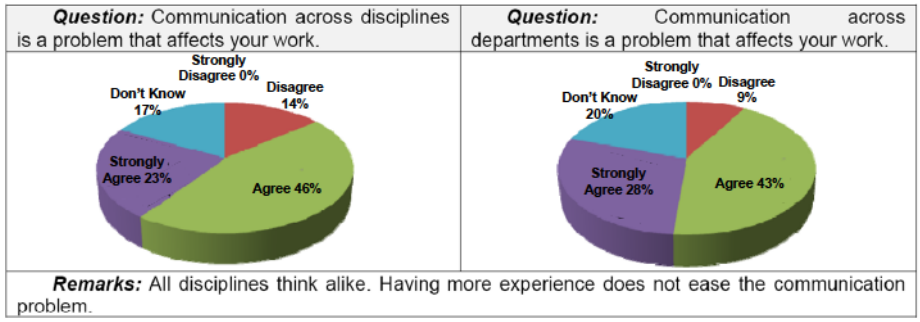
(b) Need of Overview



(c) Knowledge Sharing



(d) Finding System Information



(e) Communication Across Disciplines and Departments

Figure 3.2: Main System Evolution Barriers (details statistics per job title and experience provided in Appendix A.1)

Finding the Required System Information — Companies have plenty of ways to store and retrieve information, such as repositories, Wikis, search engines, etc. Philips Healthcare is no exception. Despite those mechanisms, as shown in Figure 3.2(d), finding system information to cope with one's work is perceived as a problem for a large part of employees. From the survey it was found that managers and domain experts suffer this problem less than other employees. Managers claimed that they probably needed less system information for their work than other disciplines, while domain experts have mastered the knowledge they need over the years. It was found that designers could not find the information they need at all, and that architects and engineers had troubles as well. Architects however stated that they have developed their own ways to find the information they need without relying on tools (mostly through oral communication). Engineers and designers on the other hand, stated that they did not have any other ways to find information than by using the repositories, which did not solve their need for information. It was found that this problem was perceived as more serious for experienced people, who probably need more specific information, that is even more difficult to find.

Communicating across Disciplines and Departments — As shown in Figure 3.2(e), it was found that communication across disciplines and departments is a serious problem that affects employees' work. We found that this problem affected all disciplines alike, regardless the experience they have. Communication problems ranged from lack of communication among different teams that resulted in integration problems, to difficulties understanding other disciplines and departments points of view (e.g. documents used to communicate) that resulted in misunderstandings and/or missing requirements.

Survey Conclusions

With the survey we showed a clear link between evolution barriers, especially ineffective knowledge sharing, to development problems and poor decisions. The reaction to the survey findings by Philips' management was, *"we have not discovered anything new; however it is clear that the magnitude of the issue is bigger than we thought, and consequently deserves more attention"*.

To find out whether those findings were Philips Healthcare specific or apply to other companies as well, they were presented and discussed with representatives from other companies such as ASML, Océ, Philips Lighting and Daimler. Although they did not provide data to support the findings, they all stated that those problems have been recognized in their own companies.

3.4 *Architectures as a Means to Understand System Evolution*

As discussed in Section 2.3.3, during the evolution process, technologies, designs and implementations may have changed completely over the years. How can we understand and support the design and evolution of a complex system, if the technology, implementation, design and even the people might have changed? The answer lies in the architecture. While implementations of the system may change several times during the lifespan of the system, it is likely that the architecture remains from the original system.

3.4.1 The Role of Architectures in the Design and Evolution of Systems

Systems Engineering focuses on a number of abstract concepts because they provide a general framework for guiding the development of systems, so that these systems will provide the desired functions in the desired ways. Among these abstract concepts is that of architecture.

The definition of architecture is still open for discussion. The architecture standard IEEE Std 1471-2000 (now ISO/IEC Std 42010) defines Architecture as *"the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution"* [IEEE]. INCOSE [Group, 2000] defines it as *"the arrangement of elements and subsystems and the allocation of functions to them to meet system requirements"*. In other fields such as software, architecture is defined as *"the highest level breakdown of a system into its parts, the decisions that are hard to change... and to whatever the important stuff is"* [Fowler, 2002].

The concept of architecture is also described by some authors as *"the earliest design decisions"*, *"those aspects that are the hardest to change"* [Klusener *et al.*, 2005] or *"the set of information that defines a systems value, cost, and risk sufficiently for the purposes of a system sponsor"* [Rechtin and Maier, 2000]. This implies that the concept of architecture goes beyond the system structure to include other important concepts such as design decisions, relevant information, etc. (see Section 4.2.1).

It can be concluded from those definitions that an architecture is paramount in the development and evolution of systems, and therefore the architecture and its related knowledge are worthwhile to preserve.

3.4.2 Architectures as a Means to Support Evolution

Architectures, once consolidated in architecture representations, are considered of paramount importance to the development and evolution of systems. Architectures provides the framework in which the evolution and design of a system is performed [Crawley *et al.*, 2004]. An architecture representation is a key artifact for the early analysis of the system, it facilitates communication and understanding among stakeholders, and drives both system design and evolution [Liang *et al.*, 2009b]. Architectures enable a way to understand complex systems, to design them, to manage them, and to provide long-term rationality of decisions made early in the project [Crawley *et al.*, 2004]. Architecture representations serve as a reference artifact that can be used to share knowledge about the design and decisions that led to the current system. In [Smolander and Paivarinta, 2002] reasons for making architectures explicit (e.g. in architecture documents) at companies are examined. The study concluded that besides the traditional use as a starting point for the system design, they serve to communicate, negotiate and to capture knowledge.

As an example, in Figure 3.3, the evolution of receivers of the MRI radio frequency chain is depicted (see Figure 2.3(d)). More than 15 years of changes separate the last receiver design from the original. The design has changed (e.g. miniaturization), the technology has changed (e.g. backplane replaced with serial), the embodiment has changed, etc. However, the architecture, as presented in Figure 3.4, has remained the same. Understanding the architecture and its related knowledge that led to its current design is key to understand evolution and to support future evolution.

Bottom line is that every system has an architecture. They may be the result of a deliberate process in the design of a system, an evolution of previous designs, the result of applying standards and protocols, the addition of smaller architectures, or just an amalgamation of

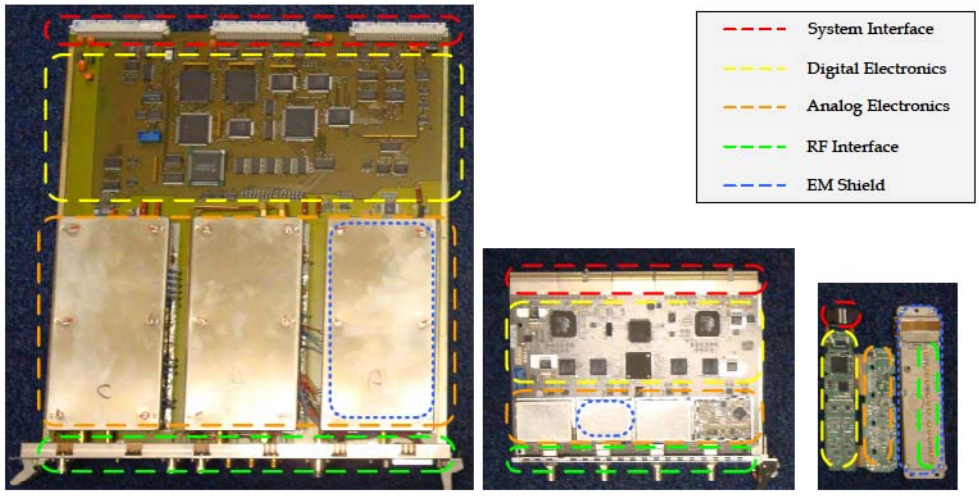


Figure 3.3: Technology Evolution (Evolution of MRI Receiver Hardware)

design artifacts. Architectures are long-lived either because they determine the design of several generations of products or because the resulting systems are themselves long-lived [Crawley *et al.*, 2004]. Architectures are therefore an adequate means to understand and manage system evolution.

Architectures are then a means to understand evolution. However, architectures being such an abstract concept, the challenge is how to represent them in a way that can be used in the development and evolution process. Other challenges are to know what information belongs to the architecture and what does not, and finally, how should architectures be captured to help overcoming evolution barriers and to support evolution.

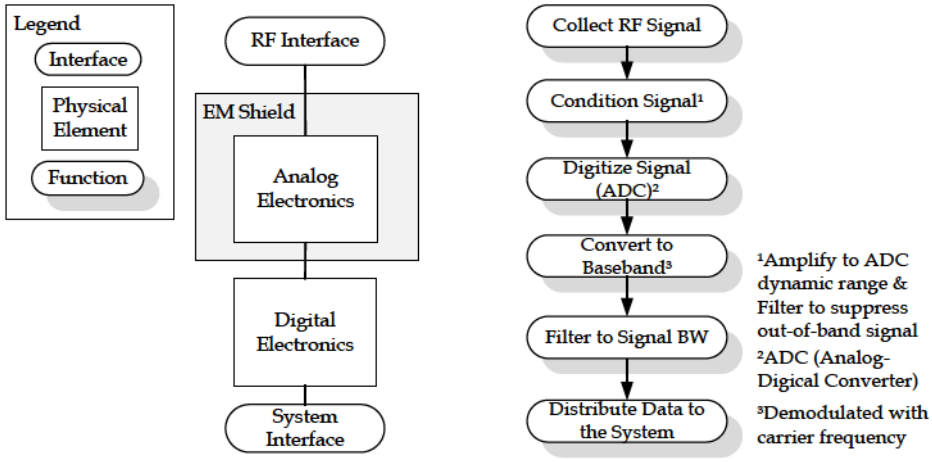


Figure 3.4: RX receiver top level architecture

3.5 Conclusions

Product manufacturers are under severe pressure to shorten development schedules, increase performance, optimize the product, incorporate more electronic and software into the product, reduce development cost and yet remain ahead of competition. As developing systems from scratch is time consuming and costly, a strategy often chosen is to evolve existing systems.

In literature, many terms refer to evolution, mostly from biological and social sciences. In other fields such as computer science, evolution has been an active field of research. In the Systems Engineering field however, system evolution is almost an unexplored field. Despite the existing contributions, an approach to deal with evolution is still needed in the Systems Engineering field.

Evolvability refers to how the system design changes from one generation to another, such as specifying which aspects of the design are passed down and which are new to previous generations. An evolvable system then would enable designers to easily identify what aspects should be incorporated in the next generation, and which aspects can be reused from the previous system to meet changing requirements, with a controlled impact of change.

Strategies to deal with system evolution are mainly based on providing design rules to add modularity to the system's design, and providing ways to estimate impact of change. Regarding modularity, the challenge is how to identify those modules, especially on systems which were not originally designed to be modular. Ideally, by estimating the impact of change, design effort could be directed towards avoiding undesired impacts. Current techniques however have clear limitations, such as the need for complete and exhaustive data.

We have found that major barriers employees face when evolving a system are; managing system complexity, lack of system overview, ineffective knowledge sharing, finding the required system information, and communication across disciplines and departments.

Successful companies have adapted their product development in ways to overcome communication barriers and facilitate collaboration. In addition, those companies are more likely to look at the system as a whole. Unsuccessful companies on the other hand, spend more resources analyzing the impact of changes, however, they lack a process to communicate changes as well as collaboration tools to make sure the changes are effectively communicated.

During evolution, technologies, implementations and designs may have changed over the years. It is likely, however, that the architecture remains. Although the concept of architecture is still open for discussion, once consolidated in an architecture representation, it is a key artifact to the development and evolution of systems.

Chapter 4

Sharing Architecture Knowledge to Support System Evolution

In this chapter architecting and known means to support it are reviewed. Duties and needs of architects are discussed. As one of the major needs of architects is to share architecture knowledge, the concept of architecture knowledge is introduced, and existing means and research approaches to share this knowledge are discussed. Finally, the impact of ineffective knowledge sharing and the need to tailor the knowledge sharing mechanism to the architecting process are presented.

In the previous chapter, development and evolution barriers were identified. Architectures, unlike technologies or implementations, are usually long-lived and once consolidated in an architecture representation, are a key artifact to support development and evolution. Consequently architectures were proposed as a way to deal with system evolution.

The process of creating architectures and its related activities is called architecting [Rechtin and Maier, 2000]. Architecting is an essential step in the design and production of complex systems [Bonnema, 2008, Gulatti and Eppinger, 1996, Muller, 2004]. The architecting process is the responsibility of an architect, and his role, duties, and skills is a common topic of discussion [Fowler, 2003, Muller, 2007a]. During the architecting process major decisions are made. For that, architects need to balance all kind of inputs, and based on their judgment, architects are expected to take good design decisions. For the decision making process, architects share ideas and knowledge with various stakeholders and collaborate with them to find an optimal solution.

Despite the importance of the architecting process, currently there is little support to systems architecting [Muller, 2004]. The process is usually performed by people who have gained experience over the years. Architecture principles, methods and frameworks aim to support this process. While architecture principles, methods and frameworks aim to provide guidance in the creation of architectures and how to represent them, there is little support on how to share the architecture knowledge generated in the process. Making architecture knowledge explicit is not common practice in most companies [Tang *et al.*, 2006]. As a result, this knowledge remains implicit in people's minds and cannot be effectively reused to design new systems.

In the following sections we review architecting and known means to support it. The duties and needs of the architect are discussed. As one of the major needs of architects is to share architecture knowledge, we review the concept of architecture knowledge and existing means and research approaches to share this knowledge.

4.1 *Architecting*

The process of creating architectures and its related activities is often called **architecting** [Rechtin and Maier, 2000]. Architecting is an essential step in the design and evolution of complex systems [Bonnema, 2008, Gulatti and Eppinger, 1996, Muller, 2004]. In this phase of

the design process, the needs and concerns of all stakeholders are taken into account to come with a well-balanced solution [Van Vliet, 2000]. This step involves, among other activities (see Section 4.1.2), determining what the system is supposed to do and how specifically it will do it.

Traditionally the process of creating an architecture follows a process of decomposition, in which a top-level concept of the system's required functions is broken down into subfunctions. At the same time the abstract version of the physical form is broken down into subsystems capable of delivering those subfunctions. This process of decomposition may continue until single parts are reached. The design process does not always follow this top-down approach but may stop when other disciplines take over the design process.

The result of the architecting phase is a series of major design decisions that shape the system to be delivered, as well as the development process [Koning, 2008]. Those design decisions are usually captured in the form of architectures, and represented in an architecture representation (see Section 4.1.1).

In this section we review existing support to architecting and the role and duties of the person in charge of this process; the architect. Finally, what the architect needs to better use his abilities during architecting is discussed.

4.1.1 Architecting Support

A distinction between engineering and architecting, is that while engineering deals mostly with measureables using analytical tools derived from mathematics and hard sciences, architecting deals largely with unmeasurables using non-quantitative tools and guidelines based on practical lessons learned [Rechtin and Maier, 2000]. Architecting remains an art which is performed typically by people who have gained knowledge and experience over the years.

One of the goals of research in systems architecting is to uncover a set of principles, methods and tools that supports the architecting process and the architect to deliver an optimal system in the given context. Among those, it is worth to mention architecting principles, methods, frameworks and representations.

Architecting Principles

Architecting is guided by lessons learned through experience and observation. Given enough lessons, their meaning can be codified into expressions called "heuristics", a Greek term for guide. Heuristics used to support architecting are usually referred as architecting principles. They encapsulate useful insights to guide the architecting process, and are meant to pass on to future generations lessons learned from experience. An example of a heuristic is:

> In partitioning, choose the elements so that they are as independent as possible; that is, elements with low external complexity and high internal complexity. [Rechtin and Maier, 2000]

Heuristics can be used either as evocative guidelines, as codifications of experience, or just as "rules" to follow in the design and development process. A collection of architecting principles or heuristics can be found in [Rechtin and Maier, 2000].

Architecting Methods

An architecting method aims to support the architecting process by providing a set of steps to guide the architect to a near optimal design. Inherited from the software field, an architecting

method that provides a stepwise process and a prescribed set of artifacts to create an architecture design is **S**tructured **A**nalysis and **S**ystems **D**esign (SASD) [Yourdon, 1989]. Structured analysis is used to analyze the user requirements and produce a structured model of the system to be developed. SASD has been around for over 20 years, and it is still popular today and widely used by many software development companies. Some widely-used architecting methods are described in [Muller, 2010b].

Architecture Frameworks

An architecture framework aims to provide guidance for what information, what presentation and what structure to use to capture the architecture. There are many architecture frameworks available, such as [Zachman, 1987], which presents the architecture design in 36 (6x6) views and provides guidance to the creation of those views. In [Greefhorst *et al.*, 2006] an overview of existing architecture frameworks is provided.

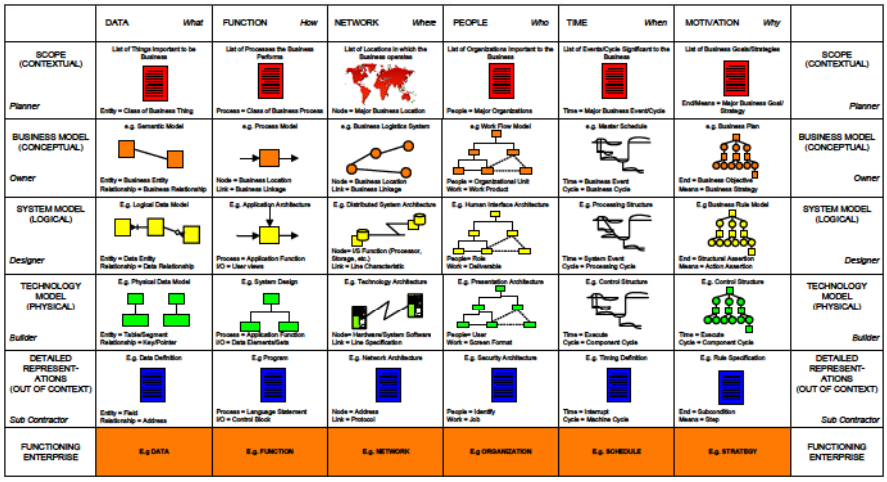


Figure 4.1: Zachman Architecture Framework [Zachman, 1987]

Architecture Representations

An architecture representation -or architecture description- defines a domain model for architecture concepts that can be used to describe architectures. There are generic standard models for architecture descriptions, such as IEEE 1471-2000 (now ISO/IEC 42010) [IEEE]. Another well-known model for architecture description is the **C**ustomer objectives, **A**pplication, **F**unctional, **C**onceptual and **R**ealization (CAFCR) model [Muller, 2004]. As shown in Figure 4.2, CAFCR is a representation of an architecture into five views:

- The Customers objective view: *what* does the customer want to achieve.
- The Application view: *how* does the customer realize his goals.
- The Functional view: which describes the *what* of the product.
- The Conceptual and Realization views: describe the *how* of the product.

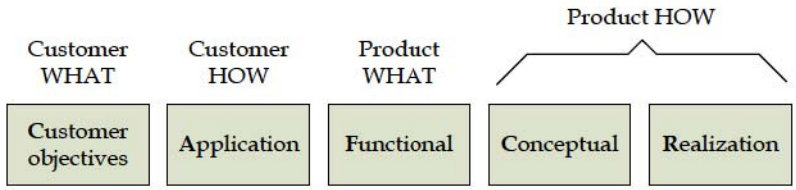


Figure 4.2: Architecture Representation in CAFCR model

Those views describe the system and its context from different perspectives such as decomposition of the system into building blocks, mapping of software into hardware elements, etc. Such views typically consist in one or more diagrams, tables or pieces of text.

4.1.2 The Architect

The responsibility of architecting lies with the architect. The role of the architect is a common topic of discussion, as well as which duties, skills and knowledge architects need [Fowler, 2003, Muller, 2007a]. Besides architecting, the architect has plenty of responsibilities and duties [Farenhorst *et al.*, 2009]. A summary of the main duties of the architect is presented in Table 4.1. Some of those duties are:

- **Decision making**: Based on his judgment, the architect is expected to balance all kind of constraints and take the best design decisions that ensure meeting stakeholder requirements. Architects are also responsible to decide an adequate system decomposition for later system integration, to map organization structure, and to estimate the impact that changes might have on the architecture.

- **Communication:** During the architecting process architects need to communicate and share ideas with various stakeholders in oder to incorporate feedback, share knowledge, inform of decisions taken, and to provide overview of the system and its context. Stakeholder collaboration in the design process is essential to ensure optimal global solution and prevent focus just on localized optimizations.

- **Documentation:** Another important activity of architects is to document the knowledge generated during the process, such as architectures and design decisions and their rationale. In addition they must ensure the integrity of the system and design specifications as they will be used later during product development. They are also responsible for the storage of architectural knowledge.

- **Asses and review:** Architects are expected to assess and review any architectural-related issues that arise in the architecting process, such as evaluate proposals, judge which of them should be implemented, and convince the stakeholders of the value of architectural decisions taken.

- **Knowledge acquisition:** Architects need to acquire new knowledge to expand the body of knowledge of the business and get insight in the best strategy for the company, to ensure that the system meets both of them. For that, they need to keep up-to-date with technology advances, research on the architecting field, learn from colleagues, etc.

Table 4.1: Architect Duties

Architect Duties
Decision Making
Balance system properties and internal design properties
Ensure meeting stakeholders requirements
Decomposition and integration
Impact of changes in the architecture
Study reasoning behind taken design decisions
Communication
Inform stakeholders from the results of the work
Obtain feedback from stakeholders
Convince stakeholders of the value of a architectural decision
Explain architectural principles
Discuss and share knowledge
Provide overview of the system and its context
Documentation
Create architecture representations
Check integrity of system specifications and designs
Reuse existing material to create new deliverables
Store architecture knowledge
Write documents such as reports and proposals for different stakeholders
Assess and Review
Evaluate architecture issues
Check and evaluate architectural proposals
Judge whether architectural proposals should be implemented
Encourage stakeholders to share their knowledge
Knowledge Acquisition
Learn from colleagues more about architectural principles
Read literature on architectures
Keep up-to-date with technology advances
Expand the knowledge about business and company strategy

The Needs of Architects

What kind of support architects need in their duties (see Table 4.1) has been addressed by some researchers, specially from the software architecting field. To support storage of knowledge, various researchers have proposed tools to help storing design decisions and rationale by using software templates, and storage facilities [Babar and Gorton, 2007, Capilla *et al.*, 2007, Jansen *et al.*, 2007]. To support decision making, research has focused on efficient search through repositories and databases [Fan *et al.*, 2006], to provide "intelligent advice" to architects [Garlan and Schmerl, 2007] and to develop tools to support writing architectural documents and perform automated analysis of the data [Liang *et al.*, 2009a]. An extensive list of available architecting tools can be found in [Farenhorst *et al.*, 2009].

Research has shown that most approaches are to support the architect duties developed from a technology perspective, which is often not the preferred solution for architects [Farenhorst *et al.*, 2007b]. Architects rather stay in control of the architecting process and are not interested in automated or intelligent support [Huysman and de Wit, 2004]. In the work of [Farenhorst *et al.*, 2009], around 300 software architects were interviewed about their activities in order to characterize architects' preferences. In this study it was found that architects do not want automatic tooling or automated support of their architecting activities. For architects,

dedicated or custom-made tools have little added functionality to support their architecting work yet require a to learn the tool and to maintain the tool. As a result, they often turn to generic tools such as modeling tools (e.g MS Visio) and spreadsheets (e.g. MS Excel).

Table 4.2: Needs of Architects (Source [Haveman, 2009])

Needs of Architects
Deliver the right information to the stakeholders while keeping the irrelevant part of information low
Make sure that information is conveyed and interpreted correctly
Collect (filter and select) relevant architecture information that exist in the organization in a smart way
Record changes in the architecture knowledge repository
Retrieve architecture knowledge stored in the heads of people
Reuse knowledge from previous experiences and products in current developments
Keep a structured overview of what has been communicated with a stakeholder

Most research in this field has focused on software architects. In order to understand what system architects need to support their duties, [Haveman, 2009] conducted a series of interviews with system architects. System architects from different companies such as Philips Cardio/Vascular, Océ, Thales and Philips Consumer Lifestyle were interviewed. The goal of that study was to determine which support system architects need. As shown in Table 4.2, it was found that main needs of architects are related to sharing and communicating architecture knowledge. Architects need to recover architecture knowledge, and to be sure that knowledge is shared and communicated effectively to the stakeholders. Architects know that reusing knowledge is essential for developing new products.

4.2 *Sharing Architecture Knowledge*

Although sharing knowledge is not common practice in most companies [Tang *et al.*, 2006], in fields such as software engineering it is considered a dominant factor for project success [Jansen and Bosch, 2005]. In Section 3.3.2 we already identified the lack of knowledge sharing as one of the evolution barriers that causes development problems and poor decisions.

One of the problems of architecture knowledge sharing is to know what information belongs to the architecture. Another problem is the consolidation of that information so knowledge can be preserved. According to [Nonaka and Takeuchi, 1995], knowledge[1] is created only when tacit information is made explicit. Therefore, architecture knowledge is created when facts and information that belongs to the architecture are made explicit (e.g. in the form of an architecture representation). In this sense, if architecture information is not made explicit (e.g. in an architecture representation), no knowledge sharing can occur.

In this section, the concept of architecture knowledge is reviewed and common approaches to share it are discussed. Finally, in order to develop a successful mechanism to share architecture knowledge, the architecting process is studied so the mechanism can be tailored to it.

4.2.1 Architecture Knowledge

It is not clear what knowledge is required to support architecting in the evolution and design of systems. In the work of [de Boer and Farenhorst, 2008], 115 papers that use the term

[1] Oxford Dictionary defines knowledge as: *what is known in a particular field or in total; facts and information*

"architecture knowledge" were reviewed. From that work it was concluded that there is not a clear definition of architecture knowledge and what type of information belongs to it. Different communities have different knowledge needs for their own purposes:

- The **requirements engineering community** sees architecture knowledge as the information that clarifies the relation between problem domain and solution space. Architecture knowledge in this sense is the knowledge that supports the rationale of the system requirements for the proposed design.
- The **model-oriented community** considers architecture knowledge as the information related to the architecture in terms of components and interfaces.
- The **software architecture community** has a view on architecture knowledge in the form of components and interfaces as well, however this community also consider design decisions that led to the design part of that knowledge.
- The **mechanical engineering community**, besides components and interfaces, this community considers the functionality of the system as a vital part of the architecture knowledge.
- The **systems design community** expands the concept of architecture knowledge to include organizations, processes, operations, people, services and technologies.

In the article of [Fowler, 2003], it is argued that the knowledge that should be part of any architecture representation is not dictated by a set of rules, but by *what is important for the stakeholder*. In other words, if a stakeholder believes something is important, that knowledge belongs to the architecture knowledge.

Although different communities and disciplines have a special interest in specific types of information, all those types identified are needed for all of them to get the complete picture of the architecture. Then, by gathering those types of information, we see that the body of architecture knowledge should at least contain, as shown in Figure 4.3:

1.- Information about the architecture structure, in the sense of components and interfaces.

2.- Information about the desired functionality.

3.- Design decisions and their rationale.

4.- Information about the problem and solution domain.

5.- Important stakeholders' concerns.

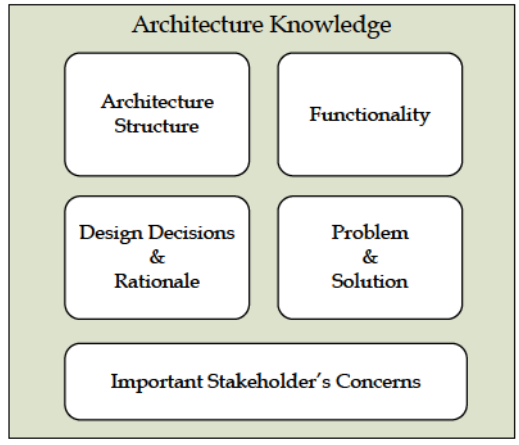


Figure 4.3: Architecture Knowledge

By looking at literature and other communities, we do not find other types of information that should be part of the architecture knowledge. Other authors of the architecture knowledge research field give other generic definitions: In [Lago *et al.*, 2008], it is argued that architecture knowledge consist in all knowledge used or produced during architecting. In [Liang *et al.*, 2009a], it is stated that architecture knowledge consist on architectures, design decisions, rationale and all other architecture relevant information. By providing the above mentioned types of information, we can ensure to meet the architecture information needs of most communities and stakeholders involved in architecting.

4.2.2 Architecture Knowledge Sharing Support

As architecture knowledge is a relatively new concept, there are only a few approaches proposed to share architecture knowledge. Among those approaches we mostly find architecture knowledge sharing through models, pattern structures and web technologies [Parizi and Ghani, 2008]. In [Liang *et al.*, 2009a] other strategies used by the software community to share architecture knowledge such as Wikis are described. Despite those contributions, according to [Avgeriou *et al.*, 2007], those approaches fail in sharing architecture knowledge as they do not provide key information such as design decisions and rationale. In addition to the lack of vital information, research effort focuses on capturing and making information available, but little research is done on how to make existing implicit information explicit, and how is that captured information consumed or communicated. A review of existing approaches for knowledge sharing can be found in [Cummings, 2003]. Research has shown that development of methods and tools alone to share knowledge does not automatically lead to increase in knowledge sharing [Ghosh, 2004], therefore, other aspects besides which tools can be developed with existing technologies should be also taken into account.

For those reasons, current practice is that architects, for good reasons, make their own choices on how to share knowledge [Koning and Vliet, 2006]. Traditional documents such as architecture documents, design specifications, reports, and similar means are still a common way to capture and share knowledge by architects [Muller, 2007b].

Text Documents as Means to Share architecture Knowledge

Within a company, text documents are a common way to capture knowledge[2]. In a company that develops complex products, it is likely that among those documents there is a SDS. A **S**ystem **D**esign **S**pecification (SDS) is used by companies in the development process to consolidate design specifications, to support 'development memory' and for educational purposes. In other words, it is the main description of a system design.

Within Philips Healthcare MRI a SDS is meant to specify how requirements are met. It also serves to consolidate the partitioning of the system into system components, to map requirements onto system elements, to define interfaces between system components, to provide budgets among components when they need to cooperate, to describe behavior, etc. In the SDS we can find most types of information that belongs to the architecture knowledge. In the Philips MRI SDS, we find the description of the system's structure in terms of components and interfaces, description of the desired functionality, some design decisions, and some additional important information.

[2]These findings are published in paper written by the author in the proceedings of *8th Annual Conference on Systems Engineering Research*, 2010. See [Borches and Bonnema, 2010b]

After discussions with several companies about the SDS, we found out that although they may vary on structure from company to company, most of them have one thing in common; they are large text documents with few drawings in them.

To assess the effectiveness of text documents such as the SDS as a way to share knowledge among multidisciplinary teams, a survey (see Section 3.3.2) was used to collect feedback from Philips employees (survey results are available in Appendix A). Some of the main results of this survey are:

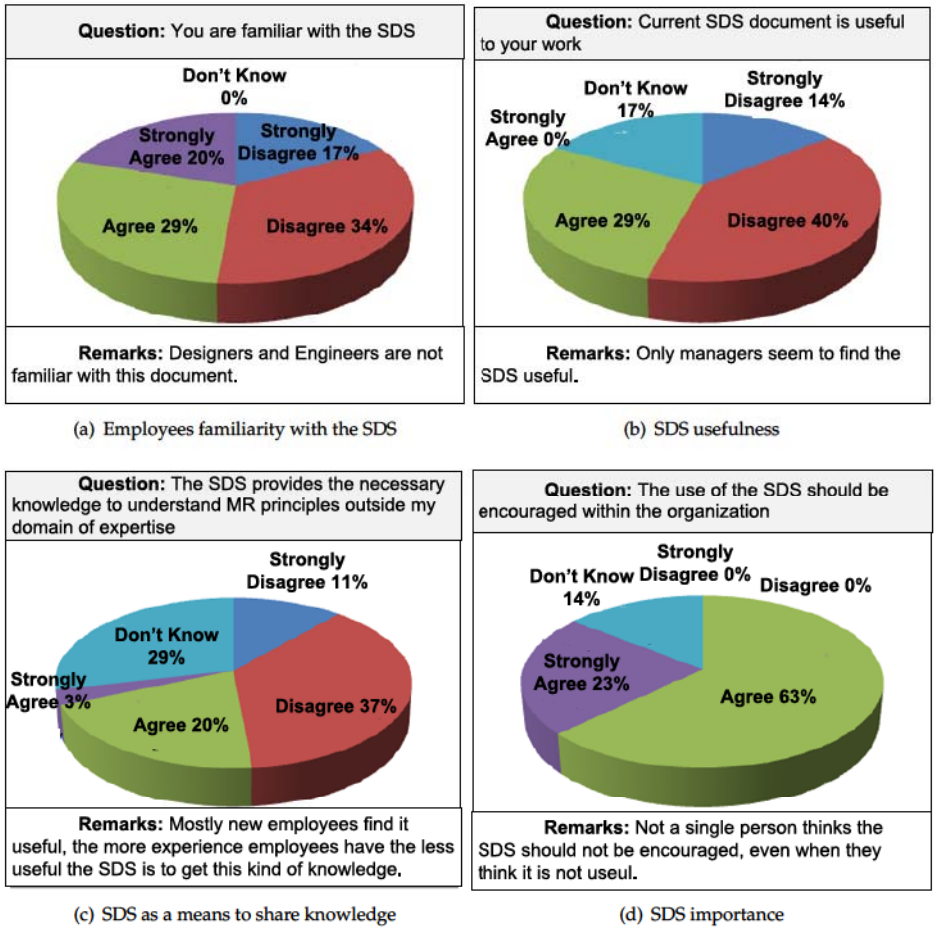
- **SDS use:** Capturing the design of a complex system such as an MRI is difficult. It requires a large and complicated SDS document (see Section 10.1.2 for details on the Philips MRI SDS). Despite the great effort invested to update and maintain the SDS document for different system releases, it was found from the survey that it was barely used within Philips. Reasons for this may be that, as shown in Figure 4.4(a), only half of the development population was familiar with the SDS document, and that only a smaller part found it useful.
- **SDS acceptance:** Architects, designers and engineers are the expected users of the SDS, however, while architects were familiar with the SDS, designers and engineers were not. The main users of the SDS were found to be managers which, as shown in Figure 4.4(b), see the SDS a useful source to get insight on the system.
- **SDS architecture knowledge support:** Regarding whether the SDS provided enough architecture knowledge outside the user's domain of expertise, as shown in Figure 4.4(c), the majority stated it does not. The SDS was found of little use in supporting communication across disciplines and departments as well as supporting management of system complexity. SDS was found useful only for new employees as a means to learn about the system.
- **SDS importance:** Despite the lack of use, as shown in Figure 4.4(d), the SDS was perceived by all employees as very important to support the development process. Most employees stated that a new format for the SDS would probably increase its acceptance within the organization.

From the analysis of the survey it was concluded that current SDS was not an effective way to share architecture knowledge, and many stakeholders stated that a new format would be desirable (see Section 10.1.2 for the new SDS style project). These results were shared with representatives of other companies to evaluate whether this was a problem of the Philips Healthcare MRI SDS. Most representatives stated that this problems are also present in their own SDS documents (or similar means).

4.2.3 The Impact of Ineffective Knowledge Sharing

As stated in Section 3.3.2, ineffective knowledge sharing results in development problems and poor decisions. In addition, research has shown that the lack of an effective mechanism to share architecture knowledge leads to other problems such as [Farenhorst *et al.*, 2007a]:

- *Dispersion of architecture knowledge*: There is no alignment between the architecture documents and the functional design and technical design documents used by developers. Because of the lack of alignment, valuable architecture knowledge might be dispersed within the organization without architects knowing it, resulting in potential integration problems.



(a) Employees familiarity with the SDS

(b) SDS usefulness

(c) SDS as a means to share knowledge

(d) SDS importance

Figure 4.4: SDS as a means to share knowledge

- *Architect work overload*: Architects usually have to explain the architecture knowledge such as design decisions more than once. The reason for this is that decisions made in meetings, including the rationale for these decisions, are not adequately stored in the architecture document. As knowledge sometimes dissipates quickly, architects need to meet again with the developers to get this knowledge explicit at a later point in time.

- *Lack of feedback*: Developers sometimes wear the hat of the architect and also make system design decisions. However, architects may not be informed on the decisions made by the developers unless explicit meetings take place as there is no effective mechanism in place that allows developers to share what they are doing.

- *Outdated knowledge in repositories*: The architecture knowledge is usually kept in experts's minds, consequently the repository contains little to no information on key aspects of the system such as the design decisions, technology preferences, standards used, etc.

As a consequence, knowledge dissipates over time, specially when someone leaves the company.

It is therefore in best interest of the company to prevent those problems by adopting and implementing an effective mechanism to share architecture knowledge.

4.2.4 Tailoring the Architecture Knowledge Sharing Mechanism to the Architecting Process

To develop a successful architecture knowledge sharing mechanism, not only does it have to meet architect's needs (see Section 4.1.2), it also has to be tailored to the architecting process. It is then important to understand the characteristics of the architecting process. When developing an architecture knowledge sharing mechanism, the following characteristics of the architecting process should be taken into account:

- **Architecting is an art**: During the architecting process the creativity of the architect plays a crucial role [Eeles, 2006]. Architecting approaches need to take this characteristic of architecting into account and should support the architect's creativity instead of constraining it. Therefore, an architecture knowledge sharing mechanism should be flexible enough to allow architect's creativity.

- **Architecting is communication-intensive**: As stated in [Clements *et al.*, 2007], in the architecting process many stakeholders are involved. In order to make good decisions, all involved stakeholders need to obtain relevant architecture knowledge. Therefore a knowledge sharing mechanism should consolidate that knowledge in a fashion that can be used and understood by a wide variety of stakeholders.

- **Architecting is constrained by time**: In practice a heavy constraint for architecting is the available time architects have. Often, time-to-market forces architects to choose for easy and simple ways to consolidate the outcome of the architecting process. An knowledge sharing mechanism should be simple enough to make efficient use of the limited amount of time architects have.

- **Architecting impacts the complete life-cycle**: Architecture knowledge is not only relevant during the architecting phase, but during all phases of the life-cycle of the system. It is therefore important to keep architecture knowledge in a way that can be used after the architecting phase. If relevant architecture knowledge is not explicitly captured knowledge loss may be the result. Consequently, an architecture knowledge mechanism should encourage the consolidation and dissemination of knowledge [Farenhorst *et al.*, 2007b].

Finally, in order to find a suitable mechanism to share architecture knowledge, insight in how the architecture knowledge is communicated and consumed by the various stakeholders is needed. Even when architecture knowledge is shared with other stakeholders, if it cannot be effectively communicated to the stakeholders, it may render a great architecting work ineffective.

4.3 Conclusions

Architecting is an essential step in the design of complex systems. In this phase, needs and concerns of stakeholders are taken into account to come to a well-balanced solution. The result of this phase is a set of design decisions that shape the product and the building process. Unfortunately there is little support to systems architecting. It is performed by people who have gained experience over the years.

The responsibility of the architecting lies on the architect. Architects have many duties; They are in charge of decision making, communication, documentation, assess and review, and knowledge acquisition. From a study in which the needs of architects were examined, it was found that one of the major needs or architects is to retrieve and share architecture knowledge. Most approaches to share architecture knowledge however are developed from a technology perspective, which is often not the preferred solution of architects. There is little support on how to make implicit knowledge explicit.

Sharing knowledge is not common practice in most companies. One of the problems of architecture knowledge sharing is to know what information belongs to the architecture knowledge. Different communities have different needs of knowledge, and therefore have different views of what architecture knowledge is. By studying the needs of different communities, we concluded that architecture knowledge should contain at least: information about the architecture structure, information about the system functionality, design decisions and rationale, information about problem and solution domain, and important stakeholder's concerns.

As architecture knowledge sharing is a recent field of research, there are only a few approaches borrowed from the software community. Those approaches, however, focus mostly on architecture structures, and neglect other vital information such as design decisions and rationale.

Architecture documents, design specifications, reports, and similar means remain as the main mechanism to capture and share architecture knowledge. Based on a survey done at Philips Healthchare MRI, by analyzing the effectiveness of the System Design Specification as a way to share architecture knowledge, we have found that it is not an effective way to share architecture knowledge.

A successful mechanism to share architecture knowledge needs to be tailored to the architecting process. For that, it should take into account that architecting is an art, so it should support the architect's creativity. It is communication-intensive, so it should be understood by a wide variety of stakeholders. It is constrained by time, so it should not demand too much time from the architect. Finally, architecting impacts the complete life-cycle, so it should encourage the consolidation and dissemination of knowledge so this knowledge is available during the whole life of the system.

Chapter 5

Effective Communication, a Basis for Knowledge Sharing

In this chapter we investigate how architecture knowledge is communicated. To understand which factors may hamper the communication process, a well-known model is proposed to model the communication process in the systems architecting context. By using that model, sources of "architecture noise" are identified and specific measures to prevent the noise from affecting the communication process are discussed. Finally, two popular strategies to support effective communication are reviewed; Model-Based Systems Engineering (MBSE) and Toyota A3 Reports.

In the previous chapter the need for an effective sharing mechanism was discussed. However, even if a mechanism succeeds in sharing architecture knowledge -all relevant architecture knowledge is delivered to the stakeholders-, if it cannot be effectively communicated -stakeholders cannot understand it-, it may render a good architecting work ineffective.

Although communication is present in many forms in all aspects of life, communicating effectively is far from a simple task. Many people have difficulties communicating across organizational boundaries and therefore in sharing knowledge [Domb and Radeka, 2009]. There are organizational measures to reduce communication barriers such as co-location of experts from different fields in the same project, design meetings, etc. However those measures do not eliminate communication barriers such as different jargons, different points of view, and do not create a synergetic way of working [Bonnema and Borches, 2008].

Research effort in knowledge sharing focuses mostly on storage, organization and retrieval of information. Little effort is spent on how information is produced, communicated and consumed. In addition, current solutions do not take into account the human side of systems architecting. Tools do not design or evolve systems, humans do [Axelsson, 2002]. For example, unlike tools, humans may overlook vital information if the reader is overwhelmed with irrelevant information or confused by the format chosen to consolidate architecture knowledge.

The architecting process is mainly a human activity, in which effective communication between architect and stakeholders is required. The communication process is in itself an essential part of the design process, as ideas, opinions, views are exchanged to acquire the knowledge to solve a problem or to improve an existing system. Architects must rely on communication to work adequately and to produce high-quality results. That communication takes place in many formal and informal ways.

As stated in Table 4.1, communication is an important duty of architects. Only by understanding the communication process, architects are more likely to achieve their objectives of sharing knowledge, influencing attitudes, or to persuade on specific decisions, which are the most prominent reasons why organizations need to communicate [Fill, 1999].

In the following sections we investigate how architecture knowledge is communicated. To understand which factors may hamper the communication process, the well-known Shannon-Weaver model (see Figure5.1) is proposed to model the communication process in

the systems architecting context [Shannon and Weaver, 1949]. Once the sources of "architecture noise" are identified, specific measures can be taken to prevent them from affecting the communication process. Finally, two popular strategies to support effective communication; **M**odel-**B**ased **S**ystems **E**ngineering (MBSE) and Toyota A3 Reports are reviewed.

5.1 Communication in Systems Architecting

Although communication is ubiquitous, it is still difficult to define. Different fields define communication in different ways depending upon their interests. As an example, from the social science field, [Ruben, 1984] states that communication is any *"information related behavior"*. Other definitions from other fields emphasize the significance of symbols, such as *"The transmission of information, ideas, emotions and skills...by the use of symbols"* [Berelson and Steiner, 1964] and *"the transmission of information, ideas, attitudes, or emotion from one person or group to another...primarily through symbols"* [Theodorson and Theodorson, 1969].

In the context of this Thesis, communication is the process by which **individuals and teams share knowledge**. Effective communication means that individuals (or teams) at one end understand the essential aspects of the knowledge other individuals (or teams) at the other end want to share.

In this section a way to model the communication process, in order to map it to the architecting context is proposed. By using that model, major sources of "architecture noise" that may affect the communication process are identified.

5.1.1 Modeling the Communication Process

Many theories exist about the communication process, such as those described in [Laswell, 1948], [Schramm, 1954] and [Berlo, 1960] to name a few. One of the most popular theories about communication comes from the information theory field; the **Shannon-Weaver model** [Shannon and Weaver, 1949]. The Shannon-Weaver model of communication has been called the *"mother of all -communication- models"* [Woods and Hollnagel, 2005], due to its simplicity and generality. The model emphasizes that the fundamental problem of communication is that of reproducing at one point a message created at another point. This theory has been widely adopted in other fields such as social science, education, organizational analysis and psychology [Verdü, 2000]. As shown in Figure 5.1, the model proposes that all forms of communication must include five elements:

- An **information source** which produces a message or sequence of messages to be communicated.
- An **encoder** or **transmitter**, which operates on the message in some way to produce an encoded message suitable for transmission over the channel.
- A **channel**, the medium used to transmit the encoded message from transmitter to receiver.
- A **decoder** or **receiver**, which reconstructs the message from the encoded message.
- An **information destination**, which is the person (or thing) for whom the message is intended.

In all communication, the encoded message is perturbed by **noise** during transmission or reception. This means that the received message is not necessarily the same as that sent out by the information source. Effective communication mechanisms have a deep understanding of potential noise sources and have developed mechanisms to avoid or prevent them from disturbing the communication.

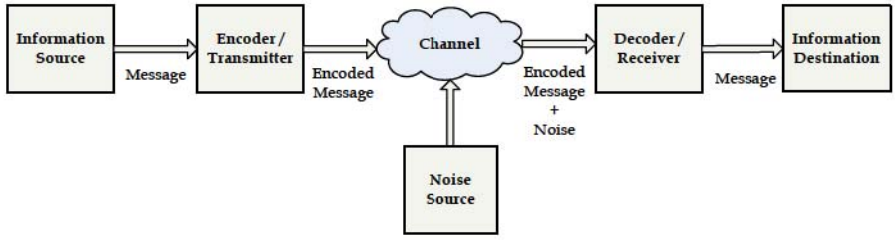


Figure 5.1: Shannon-Weaver Communication Model

If we position communication in the systems architecting context by using the Shannon-Weaver model, as presented in Table 5.1, we see the architect as the information source. The message is the architecture knowledge the architect wants to communicate. Encoding happens when the architect chooses a specific fashion to consolidate the architecture knowledge he wants to communicate and writes it down for electronic or paper distribution, or speaks it out in meetings or presentations. Architecture frameworks, modeling languages, guidelines, etc, are meant to support this encoding process (see Section 4.1.1). The encoded message is then the architecture knowledge codified in a specific fashion (e.g. views, text). The channels used are diverse, such as paper (e.g. documents), computer (e.g. email, PowerPoint) or air (e.g. oral communication). Decoding is the reading or hearing by the receivers. Finally, the information destination is the variety of stakeholders involved in the design process, such as system designers, engineers, programmers, users, managers, fellow architects, etc.

Table 5.1: Communication in the Systems Architecting Context

Architecting Element	Communication Model
Architect	Information Source
architecture Knowledge	Message
Writing / Talking	Encoder / Transmitter
Architecture Representation	Encoded Message
Paper, Computer, Air	Channel
Reading / Hearing	Decoder / Receiver
Stakeholders	Information Destination

Although the Shannon-Weaver model only takes into account noise in the channel, in architecting noises from other sources may arise, such as the context, the intentions of the architect, the choice of channel or medium, etc. In addition, matters such as the organization culture in which the message is transmitted, the assumptions made by source and receiver, their past experiences and other factors may also generate "architecture noise". For effective communication, those noise sources should be taken into account when creating the message, so the perturbation can be avoided or reduced.

5.1.2 Sources of "Architecture Noise"

As previously mentioned, the transmission and reception of messages can be distorted by different kinds of noises. Finding all possible sources of "architecture noise", that is, all factors that affect the communication of architecture knowledge, may be a herculean task. Communication involves many different domains (e.g. human perception, cognition and capability) at different levels (e.g. organization, culture, individual's background). Despite increasing interest in the impact that those human factors play in fields such as computer science (e.g. interface design), major contributions in this area are still limited [Tory and Moller, 2004]. Knowing and understanding those sources of noise would enable architects to shape messages in a way that enhances the effectiveness and likelihood of a correct reception. Although not exhaustive, we present some factors that affect the communication process (noise) that we have found relevant in our research:

Human factors — As the goal is to provide effective communication among humans, in contrast with communication among e.g. computers, it is important to identify which are the factors that may disturb the communication process between humans. Among those factors, are:

- **Limited processing capability:** Research has shown that there are clear limits to the amount of information a human can process in a period of time. In the work of [Miller, 1956], it is stated that the human brain is only capable of processing around seven pieces of information simultaneously. More information results in an overload of the processing capabilities. The result, according to [Chan, 2001], is that decision quality deteriorates under information overload. To prevent this, the information provided in a message should be limited to be within the human processing capability.

- **Variable field of view:** In the work of [Williams, 1982], it was shown that an increase in the cognitive load, it is to say, the amount of information a human must process, affects the field of view. When given a high level of cognitive load, the field of view was reduced by half. Although when a human can visualize large displays of information (e.g. a poster size), when dealing with complex information, the reader can only focus on a specific part of the information, ignoring the rest. This means that when providing complex information, the message should not be encoded in a fashion that uses the whole field of view. The needed field of view to present information should be minimized.

- **Simple decision-making capabilities:** Research has shown that there is a clear link between formalisms used to support decision making and the quality of the decisions taken [Hargis, 2000]. The efficiency of decision making is heavily dependent on the simplicity of the formalism used to display information. This is a strong driver to keep the encoding mechanism (e.g. architecture representation) simple.

- **Limits in perception:** As pointed out in [Bertin, 1983], there are clear limits in the maximum number of objects, colors, forms, sizes, etc, a user can comfortably handle. However, a viewer needs a certain minimal differentiation between types of objects, and a certain minimum space between objects. Unlike computers which can differentiate messages with one bit difference, explicit differentiation in the communication symbols is required for the receiver to understand a message. The encoding "language" and the chosen channel should take those limits into account.

- **Influence of previous experiences:** How a viewer acquires knowledge from a visual representation depends on many factors, including culture, and previous experience [Ware, 2000]. In the work of [Borchers *et al.*, 1996], it was shown that the human brain matches objects to things already known[1]. This means that a consistent way to encode the message should be applied during the communication process, in order to train the information destination in a specific way to decode messages.

There are other human factors that may affect communication, such as **semantic** noise, different interpretations of the meaning of certain words; **cultural** noise, stereotypical assumptions that cause misunderstandings; **psychological** noise, certain attitudes that make communication difficult [Berko *et al.*, 2003], etc.

Organizational Factors — Research has proven that many social [Kankanhalli *et al.*, 2005], organizational and cultural [Cummings, 2003], and personal factors [Nonaka and Takeuchi, 1995] affects communication. Among those factors, it is worth to mention:

- **Lack of a common model:** In the psychology field it is known that; *"If the concepts in the mind of one person are very different from those in the mind of the other, there is no common model of the topic and no communication"* [Taylor and Fiske, 1975]. Every person sees the world in his own way, and therefore different models are created for different stakeholders. Mental representations are therefore different, and each individual tends to believe his or her mental representation is the 'right' one. For effective communication a shared model close to the concept in the mind of the readers is needed.

- **Lack of intention:** Companies do not engage employees to perform fundamental tasks such as knowledge sharing. There is the fact that at companies a lot of the available knowledge is not made explicit at all, but instead remains tacit in the minds of people [Nonaka and Takeuchi, 1995]. The reason for this lack of intention is the belief that capturing such tacit knowledge is very hard [Haldin-Herrgard, 2000]. Information sources should then be encouraged to create more messages by providing a simple way to capture tacit knowledge.

- **Lack of autonomy:** Companies do not let employees to act independently and to find new ways to share knowledge [Nonaka and Takeuchi, 1995]. Employees are forced to use existing mechanisms and ways to share knowledge, even when they do not trust the system. Many knowledge sharing mechanisms fail because employees are reluctant to share knowledge through those systems [Kankanhalli *et al.*, 2005]. The encoding mechanism and the channel should be flexible enough to allow employees to find their ways to share their knowledge.

- **Imposition of technology-based solutions:** Literature presents warnings for the fact that incorporation of communication technologies does not mean they are automatically successful (see Section 4.2.2). Most communication or architecture knowledge is done through written means (e.g. documents) or through verbal communication (e.g. meetings). Although those means should not be discarded, as they have proven to work over the years, existing written communication means can be improved [Sobek II and Smalley, 2008].

[1]This property of the human brain is termed Law of Experience

Other organization factors may play a role in the communication process, such as the **working environment**, as how a viewer perceives knowledge in an architecture representation depends on many environmental factors including lighting conditions, visual acuity, surrounding items, color scales, etc [Ware, 2000]; the **physical distance** between the information source and destination, as research has shown that sharing knowledge is slower when company sites are further apart [Cummings, 2003]; the **relationship** between information source and destination, as duration and quality of the experience that the source and the receiver have working together influences the communication process [Cummings, 2003], etc.

5.1.3 Preventing Architecture Noise to Support Effective Communication

The aim of developing an effective communication mechanism is to strengthen the sharing of knowledge from architect to stakeholder [Koning *et al.*, 2002]. For that it is necessary to know what the architect can do when delivering a message to the stakeholders (e.g. an architecture representation), to prevent the architecture noise to disturb the communication. As shown in Figure 5.2, the communication of architecture knowledge "messages" can be divided in several general steps.

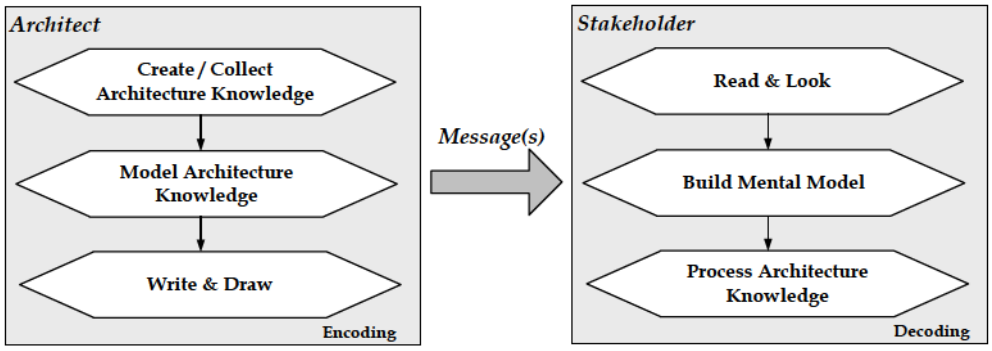


Figure 5.2: Communicating architecture Knowledge through Representations (adapted from [Koning, 2008])

On the part of the architect:

1. **Create / collect architecture knowledge:** This step consists of collecting and structuring the needed features, technologies, solutions, major design decisions, etc, that the architect wants to communicate.

2. **Model architecture knowledge:** In this step the architecture information is made concrete. In this process how the architecture will be represented for communication is decided (e.g. text, diagrams).

3. **Write and draw:** The final step is preparing the actual deliverables (message), it is to say, preparing e.g. text and diagrams using a specific notation and placing them in a specific channel (e.g. document).

In a real project these steps are not clearly divided. Actually there is a constant switching between gathering knowledge, modeling and drafting. The accent gradually shifts from the first step to the third.

Next, on the part of the stakeholder:

1. **Read and look:** In this step the text and diagrams are reviewed. Understanding the architecture starts with reading the messages. In this step the reader decides whether the information provided on the architecture representation is of any interest to him.
2. **Build a mental model:** In this step, the reader creates a mental model of the architecture from the information acquired.
3. **Process architecture knowledge:** In this step the information provided is rationally and emotionally processed so decisions can be made.

In practice there is no sharp distinction between these steps, but the focus gradually shifts from mere reading&looking to realizing the consequences. When stakeholders start to discuss with the architect both ends meet.

Table 5.2: Encoding Strategies to Prevent Architecture Noise

Human Factor	*Noise Cause*	*Communication Phase*	*Noise Prevention*
Limited processing capabilities	Information overload	Read&Look	Limit the amount of information Use visual representations
Variable field of view	High density of information, complex information	Read&Look	Use an appropriate size to display complex information
Simple decision making capabilities	Complex formalism to display information	Process architecture knowledge	Keep the formalism simple
Limits in perception	Too many visual elements and/or very similar visual elements	Read&Look	Limit the amount of visual attributes. Ensure clear differences between attributes
Influence of previous experience	Unfamiliar formalism or mechanism to display information	Process architecture knowledge	Keep a consistent way of communication. Agree on a predefined format
Organizational Factor	*Noise Cause*	*Communication Phase*	*Noise Prevention*
Lack of common model	Different views of the same concept	Build a mental model	Provide a shared view
Lack of intention	Information not made explicit, capturing information is hard	Create&Collect architecture knowledge, Write&Draw	Implement a simple process to capture implicit knowledge
Lack of autonomy	Imposition of existing mechanism, lack of trust	Model architecture knowledge	Enable a flexible way to share knowledge
Imposition of technology-based solutions	Search for technology solutions	Write&Draw	Improve existing written mechanisms

During the coding and decoding of the architecture representation used to share architecture knowledge, as shown in Figure 5.2, the message is disturbed by architecture noise, leading to communication problems. In Table 5.2 we have collected the identified noise sources and mapped them to specific steps of the coding and decoding process in order to develop simple ways to prevent the noise. Any mechanism aimed to share architecture knowledge should incorporate those prevention strategies (or similar means) so the architecture knowledge can be effectively communicated.

5.2 Strategies for Effective Communication in Product Development

In recent years, research has recognized that effective communication among employees is a key factor for project success in companies [Gruba and Al-Mahmood, 2004]. Regarding communication in the Systems Engineering field, main effort to support effective communication

has focused on the creation of models and mechanisms to support it. This is usually termed **M**odel-**B**ased **S**ystems **E**ngineering (MBSE). Another interesting strategy for effective communication comes from the management field, specifically from Toyota; the A3 Reports. A3 Reports are named after the paper size used to represent the outcome of Toyota's management system.

5.2.1 Model-based Systems Engineering (MBSE)

Model-based Systems Engineering is about elevating models to a central and governing role in the specification, design, integration, validation and operation of a system. MSBE is the formalized application of modeling to support systems design, analysis, verification and validation activities. It covers from the conceptual design phase, continuing throughout the development and later life-cycles [INCOSE-TP-2004-004-02, 2007]. MBSE is meant to be a shift from traditional document-based approaches.

Model-based Systems Engineering spans from the use of simple models to support Systems Engineering activities, to the use of formal modeling languages in a systematic way. Most disciplines today use model representations of a system to effectively communicate. In the field of Systems Engineering common forms are; Function Flows Block Diagrams (FFBD), Data Flow diagrams, N2 Charts, IDEFO diagrams, Use Case, Sequence diagrams, Behavioral diagrams, etc. Much research is done in developing new graphical metaphors [Spence, 2001]. Also is the widespread availability of software that can use model-based representations to perform automatic verification and validation of designs [Wang and Dagli, 2008].

Several factors make it interesting to use models for communication. For communication, visual modeling languages are preferred as visual representations aid readers to grasp the information better [Koning, 2008]. Visual representations not only transfer information to build up a mental model, according to [Narayanan, 1997], they also assist in processing the information. Viewing a diagram of an architecture helps keeping part of the model conscious in mind and it inspires and corrects thinking. It inspires because people can combine objects and attributes in the diagram to construct alternative diagrams. It corrects because people "see" more easily what is possible and what is not.

Modeling Languages: SysML

A current research field focuses on how to strengthen the work of describing architectures by proposing conceptual modeling languages, sometimes accompanied by software tools. A modeling language is an artificial language that can be used to express information or knowledge of systems in a structure that is defined by a consistent set of rules. There are many visual modeling languages in the field of computer science, project management and systems engineering [Wikipedia, 2010].

One modeling language that deserves special attention in the system evolution context is SysML, as it is meant specifically for systems engineering. SysML is a graphical language for building models of large-scale, complex, and multi-disciplinary systems [OMG]. There already are a number of commercial tools supporting SysML. SysML reuses a subset of UML, and adds some new diagrams specifically designed to support systems engineering. SysML is currently being intensively applied in many fields, such as verification, simulation, translation tool, etc [Kwon and McGinnis, 2007]. For effective communication however, the problem is, among other things, that these modeling languages are complicated and need long learning curve [Koning, 2008].

5.2.2 Toyota A3 Reports

Toyota Motor Corporation is perhaps best known for the efficient production system, dubbed *"lean manufacturing"* [Womack *et al.*, 1990]. Toyota's efficiency extends not only to the production floor but to product development, prototyping, testing and all other business operations [Sobek II and Smalley, 2008].

An A3 report, as shown in Figure 5.3, is a tool that Toyota uses to propose solutions to problems, give status reports and report results of information gathering activities. The A3 report is named after the paper size used to create them (297 x 420 mm, metric equivalent of 11 x 17 inches). Toyota uses the tool pervasively and it forms part of their continuous improvement program. The goal of the A3 report is to have a physical artifact that both the author and the reader, can literally point to and discuss, facilitating communication and knowledge sharing. The continuous documentation approach enables sustained organizational learning [Sobek II and Jimmerson, 2004].

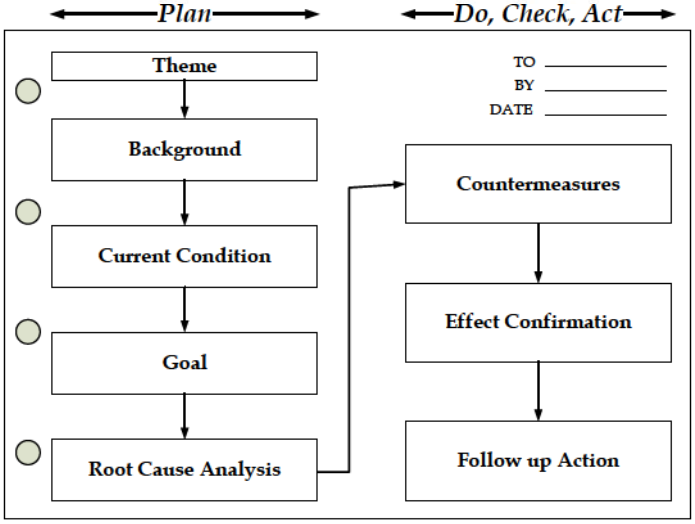


Figure 5.3: Problem Solving A3 Report (based on [Sobek II and Smalley, 2008])

The creation of A3 reports does not require long hours of specialized training [Sobek II and Jimmerson, 2004], they can be drafted with paper and pencil, enabling people to work closely to the problem without needing a computer. The graphical nature of the A3 report aims to communicate problems more clearly so readers can readily grasp the problem. The A3 size forces brevity, so synthesis of the information is encouraged. Graphical representations, such as sketches and charts are encouraged to convey information efficiently.

A3 reports are tailored to the PDCA (**P**lan-**D**o-**C**heck-**A**ct) management process of Toyota. An A3 report is a one page document that records the main results of a PDCA cycle. The PDCA cycle starts with the Plan step, in which the problem is researched until understood, finding the root causes of the problem and ideas to counteract it. The Do step is the plan in action, where measures are taken. The Check step involves measuring the effects of the implementation. The Act step refers to establishing the process, solution or system proposed if the results are satisfactory. An A3 report, as shown in Figure 5.3, establishes a concrete structure to implement the PDCA management process.

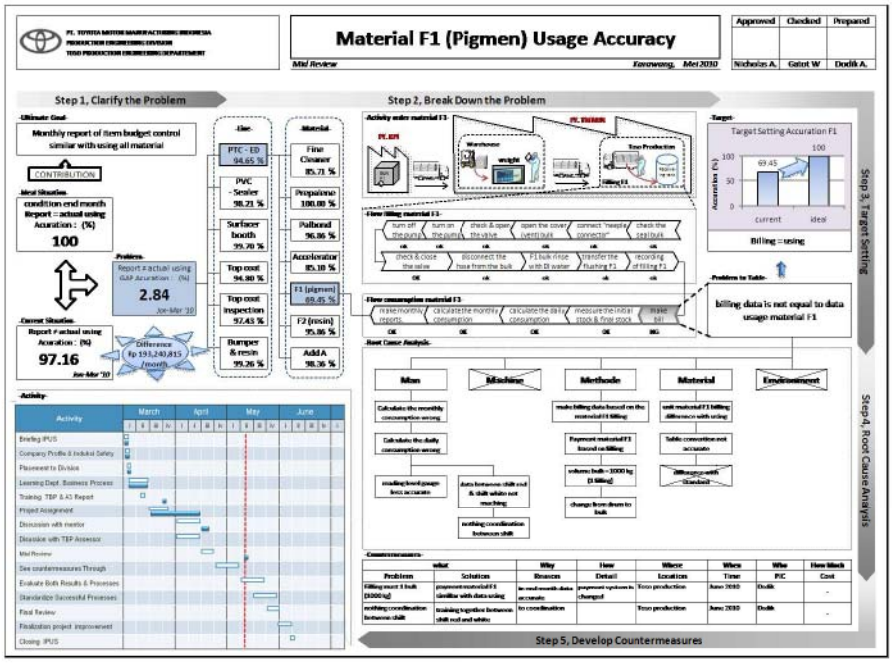


Figure 5.4: A3 Report Example (from [Ariyanto, 2010])

5.3 Conclusions

Architecting is an activity that requires effective communication between the architect and the stakeholders. Although communication is ubiquitous, it is difficult to define. In this Thesis, communication is the process by which individuals or teams share knowledge. Effective communication means that individuals or teams at one end, understand the essential aspects of the knowledge other individuals or teams at the other end want to share.

The Shannon-Weaver model is proposed to model the communication process in the systems architecting context. By using this model, sources of "architecture noise" can be identified, such as human processing limits, or lack of intention at organizations. Once main sources are identified, appropriate actions are proposed so any mechanism developed to support effective communication can prevent or minimize the noise perturbation.

The aim of developing an effective communication mechanism is to strengthen the sharing of architecture knowledge from architect to stakeholder. For that, it is necessary to know what the architect can do when creating a message to support the reader in quickly and correctly extracting the information and prevent architecture noise.

From literature, it is interesting to look at two strategies for effective communication; Model-Based Systems Engineering, and Toyota's A3 reports. MBSE elevate models to a central role in the design of a system. The use of visual models is an effective way to support effective communication, as visual representations aid readers to grasp the information better and they also assist in processing the information. An A3 report is a structured document style, tailored to Toyota's management process. The limited room to display information forces brevity and synthesis of knowledge, making it a suitable tool for effective communication.

Chapter 6

Experiences in Supporting Product Evolution in Industry

In this chapter popular approaches introduced in previous chapters are evaluated. Different study cases in which those approaches have been applied are described. The outcome of those projects and lessons learned from the experiences are presented and the results obtained are discussed. Finally, the requirements that an effective communication tool for architecture knowledge sharing must meet are provided.

The goal of this chapter is to evaluate whether popular approaches from literature introduced in previous chapters are applicable in industry. Following the *Industry-as-Laboratory* research approach (see Section 1.6), during the research carried out at Philips Healthcare, those approaches to support evolution, such as the creation of architecture representations, the use of modeling languages, and the impact of change estimation have been tested in real ongoing projects. Lessons learned from the experiences in those projects will be used to shape a more effective tool to support effective communication of architecture knowledge.

6.1 Study Cases

Following the *Industry-as-Laboratory* approach (see Section 1.6), the author ran different projects at Philips Healthcare MRI. In those projects, selected approaches were tested. These projects and approaches are presented in Table 6.1. Not all approaches from literature could be tested. Some approaches such as modularity and TRIZ (see Section 3.2) as bringing those approaches to real life situations given the time and resources allocated for the projects was unfeasible. In other cases, only well-known approaches from those available could be applied due to the limited amount of time.

Table 6.1: Projects and Approaches Used

Philips MRI Projects	**Approach Used**
MRI Communication Overview	Architecture Representation (Physical and Functional Views)
Independent Coil Releases	Architecture Representation (several views) and Specification Document
Control Communication Evolution	**D**esign **S**tructure **M**atrix (DSM), MBSE (SysML)

As shown in Figure 6.1,we selected approaches with different characteristics to evaluate whether one characteristic or another from a specific approach is more applicable in industry. We have tested approaches that are formal, it is to say, with agreed rules and conventions, and informal, in which the author can use his preferred way to display the information. We also tested free-format approaches, that is to say, those in which there are no rules to follow in the creation of the representations, and systematic approaches, which provide a specific way to the creation of the representations.

In this section those projects will be described, as well as the approaches used to meet project goals and lessons learned from the experiences.

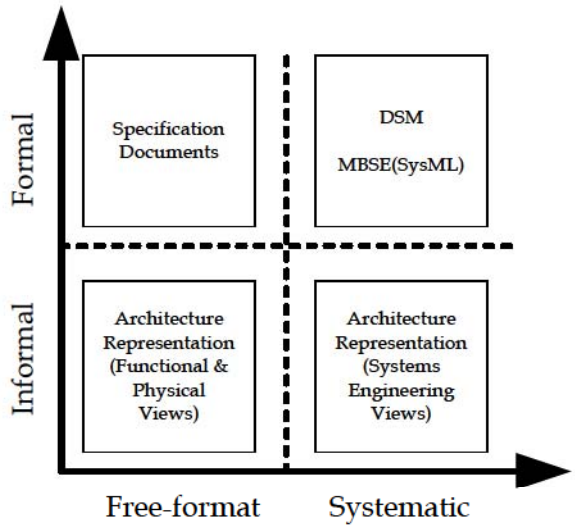


Figure 6.1: Approach Selection Criteria

6.1.1 Project: MRI Communication Overview

Like most new employees at Philips Healthcare, the author had to participate in projects without being familiar with the MRI system. Training and time to learn about the MRI system was limited due to the urgency of the projects.

In this project, a new communication design was desired for the MRI system to incorporate new functionalities and remove obsolete technology. In order to contribute to the new design, understanding of current design by the different teams that are involved in the project was essential. In a complex system such as the MRI, there are many difficulties when a change requirement has a crosscutting nature. That is, the requirement can only be met by modifying multiple chains or domains (see Section 2.2.2). This is the case for the communication subsystem, which is part of the DAS (see Section 2.2.2). Different chains have different communication requirements, and the new design must meet them.

Goal

Provide a system overview to have a better visibility and understanding of the MRI communication subsystem. Support the estimation of change propagation of a potential redesign in the DAS[1].

Approach Used

The approach chosen for this project was to create an architecture representation (see Section 4.1.1). From the many alternatives available to represent an architecture, we chose a physical building block view and an functional view to be part of the architecture representation. The reason behind this selection of views was that a building block view is a common view in any system representation, so we expected stakeholders to be able to understand it. For the

[1]The findings from this project are published in paper written by the author in the proceedings of *CIRP Design Seminar*, 2008. See [Borches and Bonnema, 2008a]

functional view, as MRI chains are building blocks grouped by functionality, therefore a way to display the functionality of the MRI chains was desired.

Therefore the approach consisted of creating a architecture representation composed of a physical view, to understand the impact of changes at the physical level, and a functional view, to understand the crosscutting nature of the changes across chains.

No systematic approach for the creation of the views was followed. The creation process was free and up to the author experience. A few pictures of a real MRI system, arranged in a hospital layout (see Section 2.3), as shown in Figure 6.2(a) were used as a starting point for the modeling process. By reviewing key documentation from the Philips MRI repository (see Section 2.2.3), a draft architecture representation was created and used as a communication tool.

To collect the implicit knowledge from the stakeholders, individual interviews were performed, mainly with system architects and system designers. A list of key architects and designers from the different chains was created. Interviewees were requested to add physical elements and functions to the physical and functional views, as well as correct inconsistencies and errors. After each interview, the outcome was validated by sending the processed information to the interviewee, who corrected and in occasions extended the captured information. This process was repeated several times (once or more per interview).

Once the views were finished and therefore the architecture representation completed (no modifications were suggested by the stakeholders), an informal way to communicate the outcome was used; a workshop with the architects and designers was organized. The objective of the workshop was to evaluate whether the architecture representation was effective to communicate and visualize the changes required in the MRI with the new DAS communication design, as well as to check whether it could be used to estimate the impact of that change.

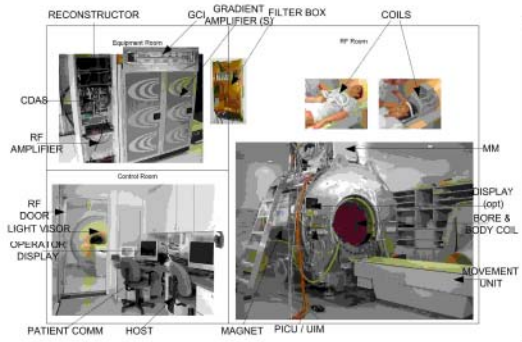
Outcome

In Figure 6.2, the views that were part of the architecture representation are shown. The **physical view** provided the physical building blocks and interfaces plotted in an A2 sheet size (420 x 594 mm) (Figure 6.2(b)), and the **functional view** provided the main functions performed by the MRI (Figure 6.2(c)), arranged horizontally by chains in an A0 sheet size (841 x 1189 mm).

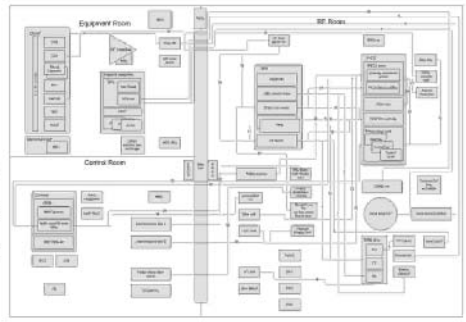
As shown in Figure 6.2(b), in the physical view, the main MRI building blocks are arranged by spatial location, and the communication interfaces displayed. The functional view, as shown in Figure 6.2(c) shows the functions required by the MRI system to create an image, from the moment the operator enters the necessary information (top of the diagram) till the image is created (bottom of the diagram). The functions are arranged horizontally by chains, and vertically according to order sequence. Functions were color-coded based on whether they are allocated to software (red) or hardware (green) building blocks. Arrows in the view represent inputs and outputs. Additional relevant information, such as quantification obtained during the workshop, was included in the views as annotations.

Creation Effort

The creation of the architecture overview took approximately 3 man-months. About 20 experts were interviewed (some of them more than once). No specific software tool was required to create the views.



(a) Philips MRI System



(b) Physical View

(c) Functional View

Figure 6.2: MRI Views (non-readable for confidentiality reasons)

Lessons Learned

The views chosen to display the MRI communication architecture (physical & functional diagrams) were considered useful by most stakeholders involved in the project. The physical view was well accepted as another MRI representation of physical elements, which is not uncommon within Philips MRI. The functional view was also accepted and reported as very interesting, even when architects and designers were not familiar to that kind of view of the MRI, they found it very useful.

The creation process was easily incorporated into daily activities and did not require specific training. The views were useful as a discussion tool, as they triggered discussions and provided people with a common framework to discuss. Employees trusted the views, as the knowledge was provided by experts within MRI. The fact that no specific tool was required for the creation of the architecture representation was important for practitioners.

The views were not easy to use due to their size. Although the views hung on the walls of some offices, they were not easy to use in daily activities such as meetings. The views ended up embedded in a Word document (as an icon, as their size was too big for an A4 sheet), as part of the architecture documentation.

When used as communication tool, quantification, requirements and additional insight obtained during the workshop was partially captured in the views. However, other informa-

tion from architecture knowledge such as design decisions and rationale was not captured by the architecture representation (see Section 4.2.1).

Some drawbacks of the architecture representation for practitioners was; lack of explicit allocation of functions into physical building blocks, limited amount of quantification data, and size of the views (as the views require a plotter instead of a regular printer).

6.1.2 Project: Independent Coil Releases

Coils are a fundamental part of the MRI system (see Figure 2.3(d)). Ideally, an MRI coil is shipped to the customer with an "information package" (e.g. CD, DVD) that can be installed on any system release. However, currently, the introduction of new coils requires adaptation of the MRI software. New coils can only be released if the customer acquires a software upgrade.

Dedicated coils are used for specific diagnosis, and coil selection determines to a large extent the quality of the images created. Customers are therefore interested in acquiring new dedicated coils during the entire life of the MRI system.

Goal

The goal of the project was to provide a way to estimate the impact that the introduction of independent coil releases would have on the system, such as affected interfaces, building blocks, etc, and the development process. In addition, the goal required to consolidate and communicate the findings to the various stakeholders involved in the project.

Approach Used

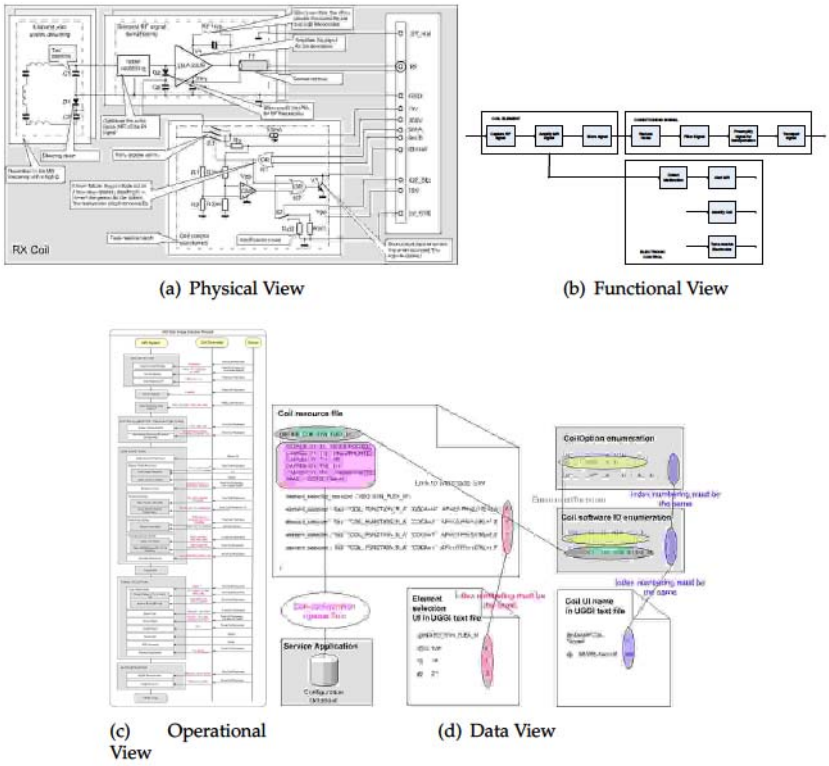
As in the previous study case, to create an architecture representation was the approach chosen for this project. Unlike the previous case, a systematic way to create specific views was followed. Specific system engineering views used for different kind of stakeholders were created in order to provide multiple viewpoints of the situation.

To collect the implicit knowledge from the stakeholders, individual interviews were performed. A life-cycle diagram of the coil was used to identify key stakeholders for each of the different phases, such as hardware, software, service, customer, application specialist, etc. Once key stakeholders were identified, meetings were scheduled with them to collect information. Views frequently used in systems engineering besides functional and physical views, such as Data Flow and Sequence Diagrams (see Section 5.2) were created for the architecture representation. The views were reviewed by different system architects.

To communicate and share the knowledge created, a specification document was created. A Specification Document was developed as a means to share the architecture knowledge as it is the formal way in which findings are consolidated and communicated within Philips MRI. In the document, views created were included and additional key information that belongs to the architectural knowledge (see Section 4.2.1) that was not present in the views such as design decisions, problem and solution, etc, was also included.

Outcome

At the end of the project, as shown in Figure 6.3, up to seven views were created with the information gathered.



(a) Physical View (b) Functional View

(c) Operational View (d) Data View

Figure 6.3: Coil View Examples

The gathered knowledge along with the architecture views, were consolidated in a 59 page Specification Document. The document contained the views, the requirements from the different stakeholders regarding Independent Coil Releases, as well as the impact the introduction of this project would have on the system.

Creation Effort

The creation of the architecture views and the specification document took approximately 2 man-months. About 16 experts were interviewed. No specific software tool was required to create the views.

Lessons Learned

In order to share the architecture knowledge, feedback from experts using the Specification Document was requested from 20 stakeholders. After 4 weeks only 3 experts found time to read the document and provide feedback. According to the stakeholders, there were mainly two reasons for this lack of feedback; firstly, the time required to read the whole document and go through all the views was too much for most stakeholders, secondly, to review a formal document such as an Specification Document required commitment from the stakeholders.

The document and the views were not easy to use. The views, although small enough to fit in a document, were seldom used or commented. The reason for that was that the views, although manageable, only provided partial views and the link among them was not clear to most stakeholders. Having more views besides the physical and functional view did not bring additional benefits. It was pointed by some practitioners that having to check more views was a burden with little additional benefit.

6.1.3 Project: Control Communication Evolution

The control communication architecture of the MRI, groups functionality in one physical place; the DAS (see Section 2.2.2). The current design of the control communication requires proprietary components and specific competences. To reduce proprietary developments and the need of specific expertise, the project aims to replace the current DAS design, C-DAS (see Figure 2.5) with a new design that uses commercial off the shelf industry standard technology.

Goal

The goal of this project was to support the evolution of the communication architecture by providing different teams with means to easily discuss alternative communication designs and requirements. In addition, it was required to provide insight regarding the impact that a redesign in the communication architecture would have on the system.

Approach Used

An MBSE approach by using SysML modeling language (see Section 5.2.1) was chosen as a way to create the different views that would belong to the architecture representation of the MRI control communication architecture. SysML guidelines were followed to model the MRI communication architecture.

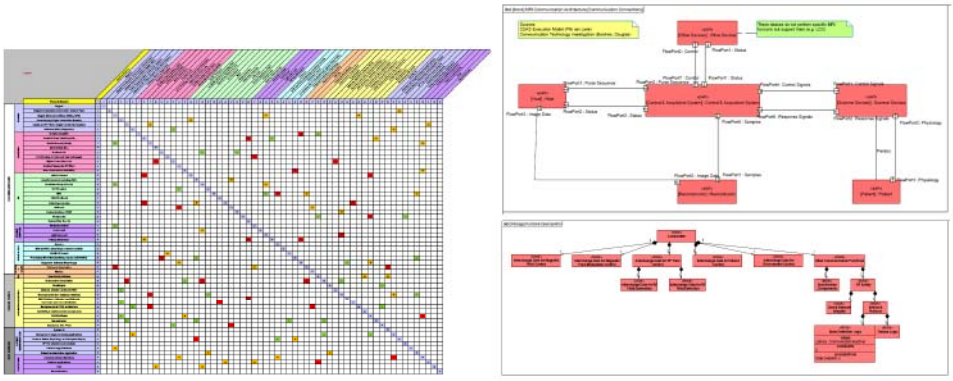
To estimate the impact of change, the Design Structure Matrix (DSM) method to predict change propagation proposed by [Clarkson *et al.*, 2004] (see Section 3.2.2) was used. The challenge of this approach was to capture the dependencies among elements of the architecture, as well as the probability that a change in one of the elements resulted in a change in the related elements in order to automate the estimation of the impact of change in the communication architecture. While the architecture representation in SysML is meant to visualize the architecture, the DSM approach is meant to provide an estimation of the impact of a design change.

As the evolution of the control communication design was mainly the responsibility of one system architect, weekly meetings were scheduled with him during the duration of the project to create the SysML views. Three other stakeholders were also interviewed for additional input and feedback on the views created.

49 building block were identified as related to the control communication. To gather the information required to fill in the DSM matrix, interface documents were reviewed, and questions to many different building block owners (see Section 2.2.3) were emailed. Email was chosen to gather the information as requesting information to fill in a 49x49 matrix (2401 cells) was unfeasible through meetings in the time allocated to the project.

Outcome

As shown in Figure 6.4(b), several SysML diagrams were created to model the control communication architecture. More than ten views were created to describe different aspects of the architecture.



(a) DSM MRI Communication Architecture (b) SysML Models MRI Communication Architecture

Figure 6.4: Techniques to Model and Estimate Impact of Change of MRI Communication Architecture

A DSM matrix of 49x49, as shown in Figure 6.4(a) was created with the main elements of the control communication architecture (49 elements). The algorithm to calculate the impact of change based on the DSM data was developed in an Excel application.

Creation Effort

The project took approximately 5 man-months in order to create the model, select appropriate SysML tools, collect data, create the DSM, etc. 4 experts were directly interviewed (some of them more than once), and more than 30 experts were requested input for the DSM matrix. A commercial software tool was required to create the SysML views.

Lessons Learned

SysML views were not successful as a way to communicate with stakeholders. Firstly, the SysML model required the creation of many small views which were difficult to visualize. Secondly, the views could not be easily shared electronically as they require a dedicated software tool for their visualization. In addition, tools that support SysML diagrams are not be currently compatible (although a standard is being developed to cope with that issue). Finally, the architect and the other stakeholders could not use the diagrams created as they were not familiar with the language[2].

Most of the time spent at design meetings using SysML was used discussing notation rather than collecting information. A common complaint from practitioners regarding the SysML views was that *"they do not resemble an MRI at all, therefore is hard to visualize the information contained in the model"*. In addition, SysML views had to be printed to be shared with the stakeholders.

[2]Although this is currently a problem, advocates of SysML claim that if SysML like UML becomes the facto standard to model systems, it is likely that future generations will be familiar with the language.

The DSM approach as a means to estimate the impact of change was unsuccessful. Firstly it took considerable amount of time to go through all the cells of the matrix (2401) to find out whether or not there was a dependency between elements, and the probability of impact. Many experts had to be consulted for this purpose and it was not easy for them to provide such information. Secondly, in addition to the difficulties gathering the DSM data, it was hard to verify the correctness of the values of the matrix. This caused stakeholders not to trust the outcome of the approach. A change in one single cell resulted in a completely different output. Finally, a major complaint was that the DSM did not allow easy visualization of the impact of change.

6.2 Discussion of the Results

An architecture representation composed by a physical and a functional view was well accepted by practitioners. Although practitioners were not familiar with functional views, they were found useful. The size of the views however (A2 and A0), the lack of explicit relation among the views, and the lack of quantification, affected the usability of the architecture representation in daily activities such as meetings. Making brief annotations in the views was the only way to capture missing architecture knowledge generated during the use of the model, additional room for textual explanations was desired.

An architecture representation composed by several views and additional architecture knowledge consolidated in a document did not lead to better results than using physical and functional views alone. Having more views did not result in better communication, as practitioners had to check more views and the additional information provided by the extra views did not lead to better insight. Using a document approach to communicate architecture knowledge led to lack of feedback from practitioners. The size of the document and the lack of time the practitioners could spend on it, resulted in poor feedback and insight.

For communication purposes, a modeling languages such as SysML resulted more a barrier than a support. As it is unlikely that most stakeholders involved in a project are familiar with the language, time is spent discussing on the notation rather than collecting the required knowledge. The need of dedicated software tools to use the modeling language led to additional communication problems, such as lack of tool compatibility, the need of practitioners to have the tool, etc.

Automated approaches to estimate the impact of change based on techniques such as DSM were unsuccessful. The need of a complete data set, the lack of impact of change visualization, and the lack of trust in the output of the approach prevented it to be used during projects.

From the experiences of applying the different approaches in industry we can conclude that, for any architect-oriented approach to be used in industry, the approach should at least meet the following criteria:

- **Small overhead:** Employees are pressured to meet deadlines, create deliverables, be aware of the time-to-market, etc. They have little time, if any, to spare learning how to use new approaches. The smaller the overhead, the more likely experts are willing to incorporate an approach to their daily activities.

- **Should not depend on custom-made software tools:** Many approaches rely on custom-made software tools to adopt the process. Although software tools do provide many

benefits such as automation, speed, verification, etc, custom-made tools require dedicated maintenance and expertise that need to be incorporated into the company. Maintenance of the tool becomes then a burden to the company. This is especially relevant if the tool is the result of a research project, in which the researcher leaves and so does the tool support.

- **Trusted output:** Many approaches rely on complex calculations to provide numbers or figures to provide guidance. Those calculations are usually hidden to the user for good reasons. Without a mechanism to provide credibility to those outputs or intensive testing of the tool to validate its outcome, users will probably not trust the results provided.
- **Should work even with incomplete and uncertain input:** In any company, especially if it is large or has different development sites across the world, finding input can be quite a challenge. Employees are used to live with incomplete information and make the most of it. Many approaches and tools however require a complete input data set to produce reliable results, which may turn in practice almost impossible.
- **Easy to use:** Meetings, discussions and other forms of communication are part of the daily activities of an architect. The approach should support the architect while performing those activities, not just when the architect is sitting behind a computer.
- **Should be "appealing":** For an approach or method to be useful, it has to attract the attention of the user. Even when the approach or method meets all the previous criteria, if it is not attractive to the user, it will be put aside or relegated to *"I'll look into it when I have time available"*.

Many approaches focus on producing complete, accurate, and reliable output. However, when some of them are brought into an industrial environment, there are some practical limitations, like those mentioned above (e.g. they depend on custom-made software tools), that prevent them to be used in practice. Unless an approach can guarantee that it will save time and money to justify the effort required to adopt it, it is unlikely that it will be used.

6.3 *Requirements for a Effective Communication Tool*

At this point, we are in a position to provide the requirements that an effective communication tool, aimed to share architecture knowledge to support product evolution, should have. By collecting the findings from the previous chapters, and the lessons learned in the application of different approaches presented in this chapter, we can define the requirements that the tool should meet. Those requirements, as presented in Table 6.2, should lead to the creation of a tool that:

- Meets practical industrial needs of communication tools, as identified in this Chapter 6.
- Meets desired properties of communication tools, as identified in Chapter 5.
- Is tailored to the architecting process, as identified in Chapter 4.
- Supports the needs of architects, as identified in Chapter 4.
- Mitigates evolution barriers, as identified in Chapter 3.
- Deals with observed evolution challenges, as observed in Chapter 2.

Table 6.2: Requirements of a Effective Communication Tool for Architecture Knowledge Sharing

Requirements
I Meet Practical Industrial Needs of Communication Tools
Require small overhead
Do not depend on custom-made software tools
Provide trusted output
Work even with incomplete input
Easy to use
Appealing
II Meet Desired Properties of Communication Tools
Provide limited amount of information
Use visual representations
Use an appropriate size to display complex information
Keep the notation simple
Limit the amount of visual attributes and ensures differences among them
Keep a consistent way of communication
Provide a shared view
Enable a flexible way to share knowledge
Improve existing written mechanisms
III Be Tailored to the Architecting Process
Support the creativity of architects
Used by a wide variety of stakeholders
Do not take too much time from architects
Encourage the dissemination of knowledge
IV Support the Needs of Architects
Deliver the right information to the stakeholders while keeping the irrelevant part of information low
Ensure that the information is conveyed and interpreted correctly
Record changes in the architecture knowledge repository
Retrieve architecture knowledge stored in the heads of people
Enable reusing knowledge from previous experiences and products in current developments
Help keeping a structured overview of what has been communicated with a stakeholder
V Mitigate Evolution Barriers
Support management of system complexity
Prevent the lack of system overview
Deal with ineffective knowledge sharing
Help finding the required system information
Support communication across disciplines and departments
VI Deal with Observed Evolution Challenges
Support moving from incremental development to top-down architecting
Reduce learning curve
Support to estimate the impact of change
Deal with the mono-disciplinary focus of developers
Support repartitioning the system

II A3 Architecture Overviews - A Tool for Effective Communication

Chapter 7

Reverse Architecting: A Process to Consolidate Architecture Knowledge

Reverse architecting aims to recover and to make explicit tacit architecture knowledge. In this chapter a reverse architecting process meant to consolidate implicit architecture knowledge in architecture overviews is designed. Challenges to the application of this process in a company are discussed, and the iterative steps required to implement the process are described. The concept of architecture overview and which views should be included are presented.

From Part I of this Thesis, we can conclude that sharing architecture knowledge is essential to support product evolution. This knowledge, however, is usually not made explicit as it is believed to be a very hard process. For this reason, it remains tacit in the minds of people (see Section 5.1.2). It is therefore no surprise that in Section 4.1.2 we found that some of the major needs of architects are to recover architecture knowledge stored in the heads of people, and to collect relevant architecture knowledge that exists in the organization in a efficient way so it can be reused in future products.

In most companies, the main architecture knowledge remains implicit in the expert's minds, and only part of that knowledge is documented. Due to this lack of knowledge consolidation, some key knowledge may be lost due to experts leaving the company, design decisions and rationale not documented, and so on. This means that architecture knowledge has to be, largely at least, recovered and made explicit. Doing this is called **reverse architecting**. How to implement a reverse architecting process is, however, unclear. The challenge is how to collect the large amount of information spread within the company, compress it to manageable proportions and present it in an easy to use way without losing essential information in the process[1].

In order to make the implicit knowledge explicit, an approach to support this process is desired. An approach designed to share architecture knowledge in a fashion that supports effective communication requires; an easy process to consolidate the implicit architecture knowledge spread within the company, and a tool that captures that architecture knowledge in a fashion that is easy to share and supports effective communication.

Based on the findings from Part I of this Thesis, and the experiences of applying different approaches in real projects at Philips Healthcare MRI, we are now in a position to design the process that, along with a tool to capture information (see Chapter 8), enables the consolidation of architecture knowledge in a way that provides effective communication.

In this chapter we propose a reverse architecting process meant to consolidate implicit architecture knowledge. The goal of this chapter is to propose a simple process that, along with a tool to capture architecture knowledge (e.g. an A3 Architecture Overview, see Chapter 8), enables to share and communicate the existing implicit architecture knowledge spread

[1]This process and experiences from its application in industry are published in paper written by the author in the proceedings of the *19th Annual Symposium of the International Council of Systems Engineering (INCOSE'09)*, 2009. See [Borches and Bonnema, 2009]

within a company. In this chapter, challenges to the application of this process in a company are discussed, and steps required to implement the process are proposed.

7.1 Reverse Architecting

Literature regarding reverse architecting is scarce. Available literature is inherited mainly from building architecture and software engineering fields. Although complex systems are usually the creation of a multidisciplinary team, lessons learned from reverse architecting in software engineering such as in [Mayrhauser *et al.*, 1999] and in building architecture such as [Galal-Edeen, 2002] may apply to the Systems Engineering discipline. In this section we review existing work regarding reverse architecting.

As stated in [Krikhaar, 1997], *"reverse architecting is a flavor of reverse engineering that concerns all activities for making existing architectures explicit, and the main goal of reverse engineering is to increase comprehensibility of the system for maintenance and new developments"*. As shown in Figure 7.1, reverse engineering is the process of analyzing a product to identify the system's elements and their interrelationships, and to create representations of the system in another form or at a higher level of abstraction that is less implementation-dependent. By reverse engineering the design of a system is recovered and made explicit [Chikofsky and Cross II, 1990].

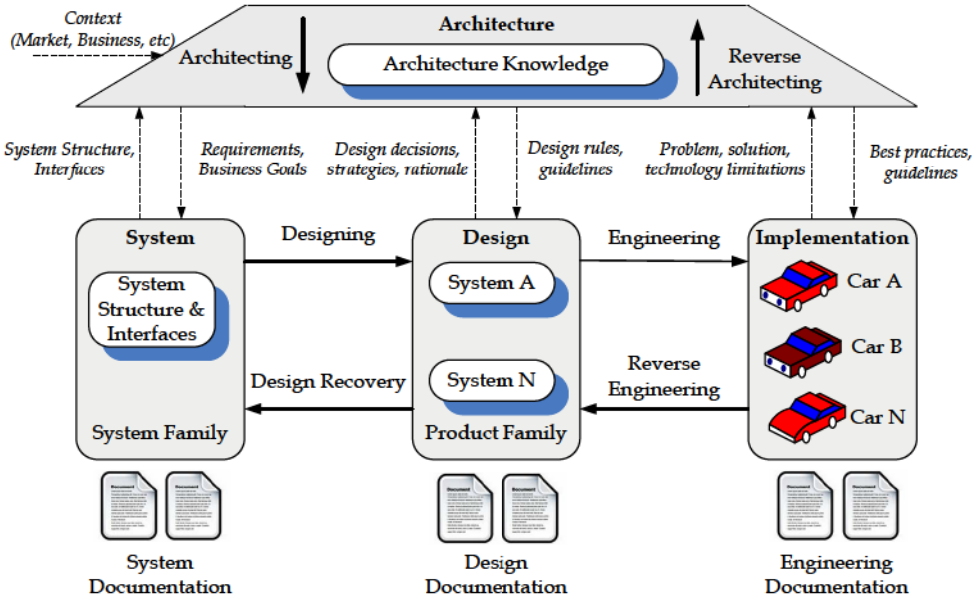


Figure 7.1: Reverse Architecting in the Development life-Cycle (adapted from [Krikhaar, 1997])

Reverse architecting, as shown in Figure 7.1 is the process that makes implicit architecture knowledge explicit. It is also a knowledge creation process. Although the process is the same as in reverse engineering, the scope of reverse architecting is different. Reverse architecting aims to recover knowledge in order to understand the system and its context, so appropriate changes can be made [Chikofsky and Cross II, 1990]. The architecture knowledge, as shown in Figure 7.1, once consolidated in an architecture representation plays a pivotal role during the complete life time of a system, guiding system development and evolution [Koning *et al.*,

2002]. While witting documents is the common approach to capture knowledge about the implementation, the design or the system (see Figure 7.1), how to capture effective architecture knowledge is still unclear (see Section 4.2.2).

7.1.1 Architecture Overview

As shown in Figure 7.2 from all the available knowledge within a company, only a small part is made explicit. Documents, diagrams, presentations and system history are the most common forms of storing this knowledge. A rich source of architecture knowledge is the employees, such as architects, experts, managers and other employees who can be interviewed to collect information.

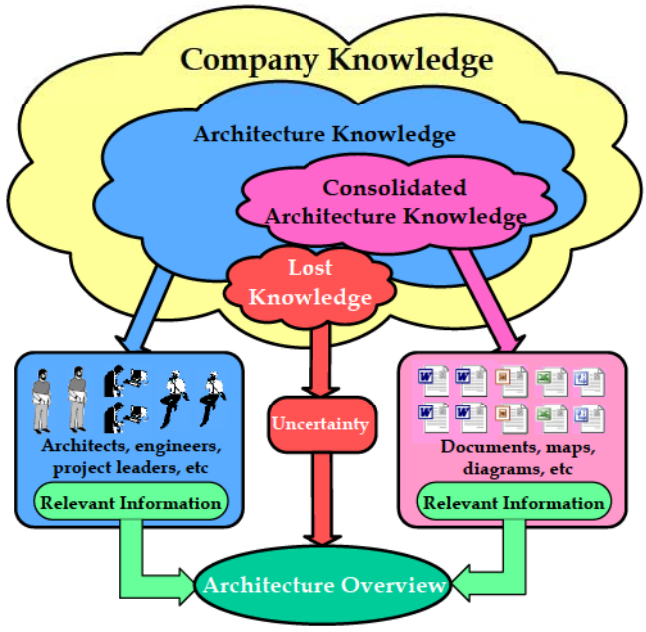


Figure 7.2: Consolidating architecture knowledge

In the reverse architecting process it is not necessary to recover all architecture knowledge. As mentioned before, architects want to deliver the right information while keeping the irrelevant information low. As discussed in Section 5.1.2, for effective communication a limited amount of architecture information should be provided; it is to say, an **overview**. In an architecture overview, as shown in Figure 7.2, only relevant architecture information is present [Bonnema and Borches, 2008]. According to [Muller, 2006], an architecture overview improves the effectiveness of the design and architecting process by:

- providing direction and guidance to the project team;
- driving and harvesting synergy by enabling better communication among disciplines;
- providing an architecture baseline and an architecture blueprint;
- capturing and sharing architectural patterns;

- giving insight in important choices, conflicts and risks;
- focusing attention in issues that go beyond physical entities.

It should be noticed that recovering the complete architecture body of knowledge is unfeasible [Ghosh, 2004] and that some knowledge may be lost or hidden, and consequently impossible to recover. Therefore, as shown in Figure 7.2 and represented by the red arrow, this missing knowledge will cause some degree of uncertainty that will be translated into any architecture representation [Muller, 2006].

Architecture Overview Views

It is difficult to determine how to present architecture information. It is not without reason that there are so many architecture frameworks (see Section 4.1.1). What is known is that a useful architecture representation should have different views of the system [Muller, 2006, Zachman, 1987], but not too many for effective communication [Parsons, 2002]. A small and representative set of views should be selected for the architecture representation.

The most common view in any architecture representation is probably a block diagram consisting of physical elements and their interfaces. For complex systems however, other high-level abstraction views that are independent of the components should also be included [Shaw, 1989]. When looking at literature regarding architectures, we find statements like:

- *"Except for good and sufficient reasons, functional and physical structuring should match"* [Rechtin and Maier, 2000].
- *"The architecture design of a system can be described from (at least) three perspectives: functional partitioning of its domain of interest, its structure, and the allocation of domain-function to that structure"* [Bass *et al.*, 2003].
- *"Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information"* [Lane, 1990].

Those and other authors make similar statements regarding the need for at least physical and functional representations of the system. From our experience in creating architecture representations (see Section 6.1.1), we have found out that those views are an effective architecture representations, however what that kind of architecture representation lacks is quantification; it is to say, numbers to grasp the relevance of the topic at hand. This finding is also present in [Koning *et al.*, 2002].

It is a well-known paradigm that you can manage what you can quantify [Gilb, 2005]. To achieve something in practice, quantification and later measurement are essential steps to ensure reaching the goal. If critical parameters are not quantified, it is less likely that people can deliver necessary performance levels. Without quantification, employees have no clear targets to work towards, and there are no precise criteria for judgment of failure or success [Gilb, 2008].

Not all architecture knowledge is captured in those views, however, additional architecture information such as design decisions, their rationale, etc, should also be part of the architecture overview. Those views should be related to each other, and mapped as much as possible one another. Finally, additional information needs to be added to those views to complete the required architecture knowledge such as design decisions and rationale. We can

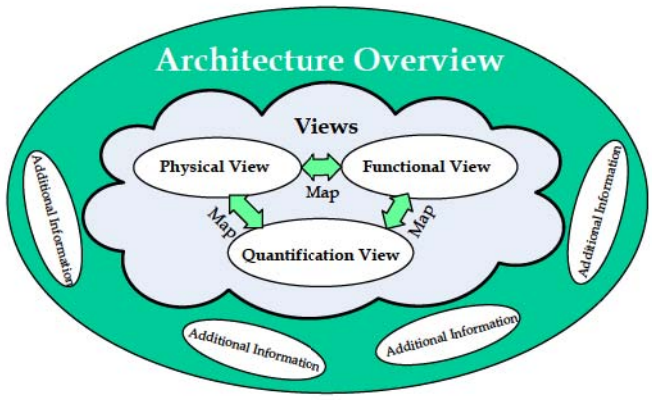


Figure 7.3: Architecture Overview: Presenting architecture Knowledge

conclude that, as shown in Figure 7.3, **a good architecture overview should have at least three interconnected views: functional, physical, and quantification**, optimally supported with **additional information**.

7.2 *Reverse Architecting Process*

In [Muller, 1996], it is mentioned that the reverse engineering process (see Section 7.1) consists of three phases; **information extraction, abstraction** and **presentation**. Although not explicitly mentioned in literature, it is clear that frequent iterations are required in order to incorporate new insights and findings discovered during the process. As shown in Figure 7.4, reverse architecting being a specific flavor of reverse engineering, the same phases apply.

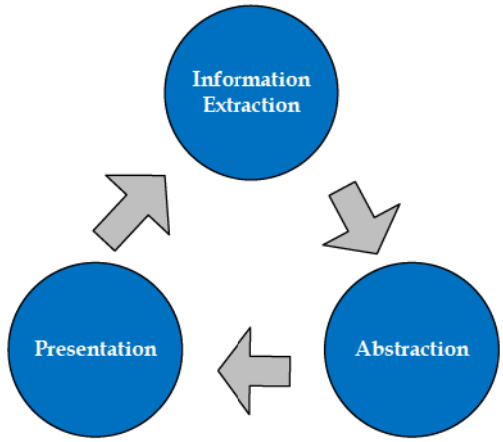


Figure 7.4: Reverse Architecting Process

Main challenges of the reverse architecting process are; how to extract the adequate information in an effective way, what information to keep and how to compress it effectively with-

out losing essential information in the process, and how to present the abstracted information in a way that supports effective communication. In the work of [Eppler, 2004] communication problems between experts and stakeholders were investigated. The study concluded that *"experts struggle with three major issues when transferring their knowledge: First, reducing or synthesizing their insights adequately, second, adapting these trimmed insights to the stakeholders' context without distorting them, and third, presenting the compressed and adapted findings in a trust building style and reacting adequately to questions and feedback."*. Those findings are in line with the findings from previous chapters, in which we found out that among major needs of architects are 4.1.2):

- Collect relevant architecture information that exist in the organization in a smart way; that is to say, how to extract information.
- Retrieve architecture knowledge stored in the heads of people; that is to say, how to extract tacit information.
- Deliver the right information to the stakeholders while keeping the irrelevant part of information low; that is to say, how to abstract.
- Make sure that information is conveyed and interpreted correctly; that is to say, how to present the abstracted information.

In this section we describe an iterative process that can be used to recover implicit or spread architecture information, and bring it together, consolidating it in an architecture overview. The process includes a series of three main steps, each of which breaks down into individual considerations explained through the section. The iterative process enables one to produce architecture overviews that can be refined by repeating the steps. At the end of the process an architecture overview should be available to communicate the architecture knowledge to all interested stakeholders.

7.2.1 Information Extraction

Information extraction consists of effectively collecting relevant architecture information from the main sources. The goal of this phase is to extract and identify information that belongs to the architecture, and once consolidated, will become part of the architecture knowledge.

Information Extraction Challenges

When extracting architecture information from within an organization, some of the challenges are:

- **Finding system information**: As stated in Section 3.3.2, one of the major problems of employees in product evolution is finding system information. Finding system information in repositories and documents is difficult [Koning *et al.*, 2002]. Finding an expert in the field seems to be the preferred option (see Section 4.2.2), yet in large organizations finding the right persons might be a challenge as well.
- **Capturing effectively the information gathered**: At meetings or discussions, people may bring booklets or similar means to take notes and collect the insight generated. White boards and similar means are common tools used during discussions. However the notes and drawings generated are rarely consolidated or shared after the meeting.

This is due to the lack of an effective and simple mechanism to capture and share those insights. This leads to the need of more meetings and discussion in the best case and misunderstandings in the worst case.

- **Understanding the information**: There is a lack of uniformity in the explicit knowledge. Different disciplines use different techniques to consolidate their knowledge such as specific languages, models, notations, etc. In addition, there are many ways in which experts provide information (such as mathematical formulas, drawings, textual explanations, etc). When gathering information from different sources, it might be not easy to understand information, leading to communication problems (see Section 3.3.2).

For any information extraction process, all those challenges should be taken into account in order to develop an effective and efficient way to avoid or minimize these effects.

Information Extraction Strategies

There are mainly two strategies to extract information; human-based and technology-based strategies. As an enormous amount of information exists only in natural language form (e.g. text documents), an area of research focuses on how to automate the information extraction from documents by filtering information from large volumes of text. Among those techniques we can find the automated retrieval of documents from collections, the tagging of particular terms in text, algorithms to structure and analyze data, etc. An overview of those techniques can be found in [Grishman, 1997].

While technology-based strategies are useful, they do not deal with how to extract implicit information stored in people's heads. In addition, those approaches are usually difficult to use. From Section 4.1.2, we learned that architects want simple and smart ways to extract this information, and that they do not like technology-based solutions that hide the information (non-transparent). To collect information that should be included as part of the architecture knowledge a simple and transparent process is desired.

Information Extraction Process

From the experiences in collecting architecture information (see Chapter 6), we have distilled the following steps to extract architecture information:

1. **Stakeholder selection:** A large amount of architecture knowledge is implicit. Therefore, extracting information from stakeholders is essential. To conduct a series of interviews and keep them manageable and productive, it is necessary to first identify a set of representative experts about the specific topic under study. Not all stakeholders can provide architecture knowledge or value it.

   On occasions, required information may be spread over the life-cycle of the system, it may be necessary to identify the life-cycle of the topic under study (e.g. from business need to disposal). Once the life-cycle is identified, key representatives of each phase of the life-cycle can be identified. Then, taking into account the relevance of each phase and the time available for the information extraction process, a selection of key stakeholders should be made. A meeting with a senior architect or project manager can be held to validate those key representatives.

2. **Preparation:** To take advantage of the expert's knowledge, it is important to become familiar with the topic under discussion. Relevant documentation should be found, reviewed and analyzed, architecture information extracted, and key questions annotated.

   A draft architecture overview (see Section 7.2.3) with the information extracted from documents or similar means should be created to consolidate and communicate the findings. In order to know whether information should be part of the architecture overview, the information type should be part of the information required for architecture knowledge; that is to say: architecture structure, design decisions and rationale, functionality, problem and solution, and important stakeholder's concerns.

   Prior to an interview, information gathered -in the form of an architecture overview- should be sent to the interviewees explaining the reason of the interview. This allows interviewees to prepare themselves.

3. **Questioning:** During an interview key questions are asked. Specific questions collected in the preparation step should be asked. The architecture overview can be used during this step as an artifact to guide the expert and extracting the desired information. For example, the question generator proposed in [Muller, 2004] can be used to generate key questions by using the architecture overview:

   How about the <characteristic> of the <component>, when performing <function>?

   The architecture overview not only enables guiding in the questioning, it also enables reviewing the architecture information gathered; it is much easier for an expert to modify a wrong view in the architecture overview than creating a new one from scratch.

   Questioning should end by asking the expert for relevant additional sources of information such as documents or further contacts.

4. **Update and validate:** The information collected should be reviewed and incorporated in the architecture overview when needed.

   When possible validation of the outcome should be made, either by e.g. measurements to the system, review by a senior architect or key stakeholders, etc. Feedback is necessary to validate the information, to clear up inconsistencies, and to create trust in the description. If several senior architects have approved of the overview, it will probably be trusted by many engineers.

The information extraction process can be implemented in many different ways, such as in face-to-face meetings, emails, remote collaboration, workshops, etc. According to [Daft and Lengel, 1984], face-to-face communication is the richest form of communication as it provides instant feedback, the capacity to transmit multiple cues such as body language, voice and inflection, and because face-to-face communication uses natural language. Face-to-face communication was the preferred option in the projects performed during this Thesis (see Chapters 6, 10).

It should be noticed that the process of information extraction has two implicit parts. First the individual facts from the sources are extracted. Second those facts are integrated, producing higher level facts or new facts through inference. This means that the reverse architecting process not only enables to recover existing knowledge but also to create new knowledge by inference.

7.2.2 ABSTRACTION

To abstract consists of filtering and grouping extracted information to obtain a manageable set of relevant information. This is an essential step to prevent information overload (see Section 5.1.2). As stated in [Eppler and Mengis, 2003] *"the essential mechanisms to fight information overload are to assure that it is of high value, that it is visualized, compressed and aggregated"*. It is to say, the main challenge of this phase is to know what information to keep (high value) and how to compress and to display it without losing essential information in the process. The goal of this phase is to extract the relevant information and to compress it in an appropriate format.

Abstraction Challenges

When abstracting a set of architecture information, some of the challenges are:

- **Surfacing key ideas:** It is not easy to identify which are the key ideas (or facts) from the information gathered. In most documentation there is a lack of stress laid on key information [Roche, 1979]. This means that the reader needs to do extra processing to extract key ideas from the documentation.
- **Aggregating information:** Individual ideas need to be related to a whole in a way that produces a harmonious whole, in order to produce larger information (e.g. through inference). Individual facts convey very little, but when pieces are fitted together properly, the picture becomes clearer. Without an idea of the whole, readers may have trouble understanding the parts.
- **Unstructured information**: The lack of a comprehensive structure, not the amount, is an important reason for inability to find relevant information [Koniger and Janowitz, 1995]. For example, within a document, we can find different types of information, such as architecture, design, implementation, rationale, etc, making hard to separate the information types. This also leads to outdated information, as documents need to be updated even when only part of the information changes, yet other information type is still valid (e.g. architecture).

For any abstraction process, all those challenges should be taken into account in order to develop an effective way to produce good overview in a reasonable amount of time.

Abstraction Strategies

Finding ways of abstracting information is common in Systems Engineering. Due to the large-scale and multidisciplinary nature of complex systems, it is necessary to develop some form of abstraction in order to be able to describe and grasp essential aspects of the system. Using visual representations in the form of models is a common way to abstract information. The design community has a long tradition of building models to represent a diverse set of systems [Bonnema and Houten, 2006]. Different research communities are intensively developing new graphical representations to best abstract and present their specific needs for information [Spence, 2001].

To support the creation of good visual representation, many researchers refer to the work of Edward R. Tufte. In his first book, *"The Visual Display of Quantitative Information"* [Tufte, 1990] Tufte focuses on charts and graphs to effectively display numerical information. His

second book, *"Envisioning Information"* [Tufte, 1997], explores similar territory but with an emphasis on maps and cartography. His third book, *"Visual Explanations"* [Tufte, 2001] centers on dynamic data; information that changes over time. Tufte has described the three books as being about, respectively, *"pictures of numbers, pictures of nouns, and pictures of verbs"*. Those resources can be used to create effective visual representations to abstract any large amount of information.

Abstraction Process

From the experiences abstracting architecture information (See Chapters 6, 10), we have distilled the following steps to abstract the architecture information:

1. **Limit the amount of information**: The smaller the amount of unnecessary information a stakeholder must consult, the better this information is [Koning and Vliet, 2006]. An optimal way to ensure that the stakeholder will only have to consult a small amount of information is by limiting the amount of information that will be provided to them. By forcing a limit in the amount of information, like in the A3 reports (see Section 5.2.2), brevity and synthesis are encouraged. Nonessential information must be removed until only relevant information remains.

2. **Filter information**: A criteria to decide what to keep from the information extracted must be developed. The value of information is not in the amount of information, but in the key insight it brings to the specific stakeholder [Simpson and Prusak, 1995]. Only what is relevant for the stakeholder about the information gathered should be kept. According to [Roche, 1979], key ideas in a text document can be identified by:

   - it starts with a statement about a situation,
   - it continues with a comment about the situation that narrows the range of implications, and
   - it concludes with a statement about the implications of the comment.

3. **Group information**: Filtered information should be labeled according to their type in order to group related information together and being able to create larger facts through inference. For this, a classification of information should be made, and filtered information assigned to one (or more) of those classes. There are two ways of grouping; as members of a class (grouping facts as if they were objects) and as a steps in a process (grouping facts as events). To create a proper grouping:

   - Define a class of information to be grouped.
   - Make clear what differentiate this group from the others.
   - Determine what is relevant for the information to belong to this group.

4. **Provide visual representations**: As stated before, an effective abstraction strategy is by using visual representations. The value of diagrams in communication very often expressed by the saying *"a diagram is worth ten thousand words"*, because they aid in recognizing complex relationships between many elements [Larkin and Simon, 1987] (examples shown in Figure 8.6). If possible, a group of information should be represented visually, in a way that is compact and easy to understand.

The abstraction phase results in interrelated pieces of key architecture information, represented in an compact way. For the abstracted information to provide an overview, those pieces of information should be presented together in a structured format that provides overview.

7.2.3 Presentation

Presenting consists of choosing the appropriate format and style to deliver the information to the appropriate audience. The way information is presented can influence, to a large extent, the message that is understood by the reader (see Section 5.1.1). An explicit presentation of information is paramount as it delivers better insight, and leads to a more complete design Koning and Vliet [2006]. The goal of this phase is to display architecture information in a format that is compact, accurate, adequate for the purpose, and easy to understand.

Presentation Challenges

When presenting architecture information, some of the challenges are:

- **Choosing text or model representations:** When presenting information it is necessary to decide whether the architecture knowledge is presented in a model or in text. Architecture standards encourage the use of models, and in our survey we found that this is also preferred by most disciplines (see Appendix A). However, even when model approaches provide clear benefits over pure textual descriptions, this does not mean that an architecture representation should provide just models. Both text and models are needed in any architecture representation. Text can be very strong in suggesting the proper interpretations and associations in the views [Koning *et al.*, 2002]. The challenge is then how to combine visual and textual representations.

- **Using an adequate language:** A challenge when presenting architecture information is what "language" should be used in the architecture representation; natural language, domain languages, or formal languages such as SysML (see Section 5.2.1). The audience of the architecture information is usually diverse, therefore domain or formal languages may be hard to understand by people which are not familiar with them. For that reason, natural language, even when more ambiguous, should be used when presenting architecture information.

- **Selecting architecture view(s):** It is difficult to determine which views to use to present architecture information. It is not without reason that there are so many architecture frameworks and representations(see Section 4.1.1). In addition to the many ways in which an architecture view can be represented, selecting the appropriate view and characteristics (size, complexity, etc) for the intended audience is also a challenge.

For any presentation process, all those challenges should be taken into account in order to develop an effective presentation strategy.

Presentation Process

From the experiences presenting architecture information (see Chapters 6, 10), we have distilled the following steps to present the architecture information:

1. **Select appropriate architecture view:** An appropriate view from the architecture overview (see Section 7.1.1) to represent the abstracted information must be selected, as well as an appropriate representation for that view; visual, textual, or a combination of both. Other aspects, such as the size of the view, the room available to display it, etc, should also be taken into account in this step.

2. **Complete information:** As there may be some important facts not captured by the architecture view, it is necessary to record those and find an appropriate way to include them in the architecture representation.

3. **Place outcome in the architecture overview template:** In order for the reader to easily find the information he is looking for, the view needs to be allocated to a predefined place in the architecture overview template (e.g. a document, a software tool, or an A3 Architecture Overview), so the reader can become familiar with the structure of information.

Once a process is available to collect, abstract and present architecture knowledge, what it is needed is a tool that captures the implicit architecture knowledge generated in a format that best supports effective communication [Koltay and István, 2009]. In the next chapter, a tool designed for effective communication of architecture knowledge in an industrial environment, the **A3 Architecture Overview**, will be presented.

7.3 *Conclusions*

Main architecture knowledge is implicit in expert's minds, and only part of that is documented. Some of this knowledge may be lost, due to experts leaving the company, design decisions not documented, etc. This means that architecture knowledge has to be, largely at least, recovered and made explicit. Doing this is called reverse architecting.

Reverse architecting is a flavor of reverse engineering that concerns all activities for making existing architectures explicit. Architecture knowledge, once consolidated in an architecture representation plays a pivotal role, guiding development and evolution. For that, an architecture overview is proposed as a way to represent architecture knowledge in a way that enables effective communication. In an architecture overview, only relevant architecture information needed to support evolution is present.

Useful architecture representations have different views of the systems, but not too many. A small set of representative views should be selected. The most common view in any architecture representation is a physical view describing building blocks and connectors. For complex systems, other views such as a functional view are also needed. A quantification view, providing relevant numbers to grasp the relevance of the issue at hand, is also needed. As not all architecture knowledge can be captured with those views alone, an architecture representation needs to provide room for additional information such as design decisions and rationale. An architecture overview should then have, a physical view, a functional view, a quantification view, and room for additional information.

The reverse architecting process consist of three iterative steps; information extraction, abstraction and presentation. The iterative process enables one to produce architecture overviews that can be refined by repeating the steps. At the end of the process an architecture overview should be available to communicate the architecture knowledge to all interested stakeholders.

Chapter 8

A3 Architecture Overviews

In this chapter the A3 Architecture Overview is designed. The goal is to design a tool to support effective communication of architecture knowledge. Requirements and lessons learned in previous chapters are used to ensure that the design enables the architect to share and to effectively communicate architecture knowledge. Different aspects of its design and their rationale are presented. Finally, how to create a repository of architecture knowledge with A3 Architecture Overviews is discussed.

An A3 Architecture Overview, as shown in Figure 8.1, is a tool designed for knowledge sharing and effective communication of architecture knowledge. An A3 Architecture Overview provides a framework in which key architecture information obtained during the reverse architecting process is consolidated in order to share architecture knowledge[1].

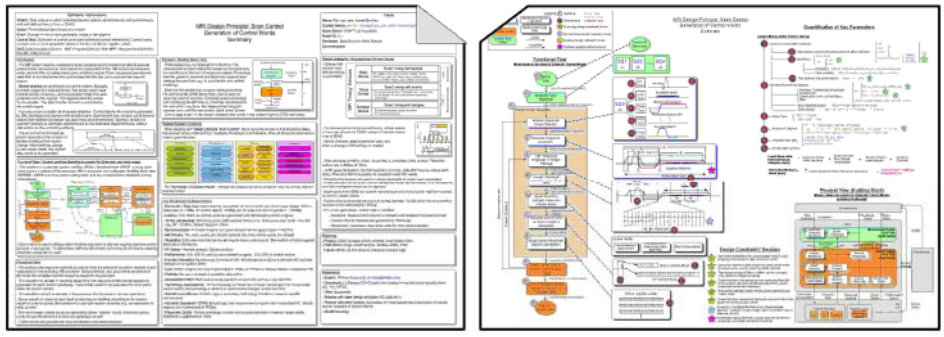


Figure 8.1: A3 Architecture Overview Example. Left: A3 Summary; Right: A3 Model

As shown in Figure 8.1, an A3 Architecture Overview uses two sides of an standard A3[2] paper size. One side displays a structured model (A3 Model), composed of several interconnected views, while the other side displays structured textual information (A3 Summary). The A3 Architecture Overview integrates multiple pieces of architecture information from different sources into a predefined structure to provide the reader a coherent picture of a system aspect.

The goal of the A3 Architecture Overview is to enable sharing of architecture knowledge by using a format that encourages its use, and to enable effective communication by providing an architecture overview in a fashion that can be understood by a wide variety of stakeholders.

To design an effective communication tool to support product evolution in an industrial environment, human and organizational factors as well as experiences in the use of other tools

[1]The A3 Architecture Overview and experiences from its application in industry are published in paper written by the author in the proceedings of the *20th Annual Symposium of the International Council of Systems Engineering (INCOSE'10)*, 2010. See [Borches and Bonnema, 2010a]

[2]A3 is an international paper size standard of 297 x 420 mm (American metric equivalent of 11 x 17 inches)

have been taken into account. The aim of an A3 Architecture Overview is not to be complete, formal or executable; an A3 Architecture Overview is meant as an artifact to communicate and share architecture knowledge.

In this chapter, we introduce the A3 Architecture Overview, and we provide the requirements and lessons learned in previous chapters so its design enables one to share and to effectively communicate architecture knowledge.

8.1 A3 Architecture Overview Objectives

The main goal of an A3 Architecture Overview is to consolidate architecture knowledge in a format that enables sharing it in a fashion that supports effective communication. By providing this tool, we aim to support architects in their activities during product evolution, and during the design process in general. To meet that goal, an A3 Architecture Overview has to meet the following objectives:

Aid in Product Evolution

By focusing the contents of the A3 Architecture Overview at the architecture level (see Section 3.4.2) the A3 Architecture Overview aims to capture key knowledge that is essential and that can be (re)used to develop new products or product evolutions.

To support evolution of complex systems, in which multidisciplinary teams are involved, the contents of the A3 Architecture Overview will be the architecture knowledge that diverse disciplines require for their work (see Section 4.2.1).

Provide a System Viewpoint

As discussed in Section 3.1.1, product evolution requires to make changes to an existing system. The ability of a company to understand the impact that changes have on the system determine their ability to cope with product evolution. For that, maintaining a system viewpoint is important in order to see how parts fit into the larger picture and to estimate the impact that a change may have in the system.

Successful companies are more likely to look at the system as a whole (see Section 3.3.1). Neglecting a system's viewpoint may cause that a problem solved locally causes another problem elsewhere. For that, by providing a system viewpoint, an A3 Architecture Overview aims to enable better understanding of the system as a whole, and to provide a way for the architect to foresee and communicate the consequences of potential changes.

Support the Architecting Process

The A3 Architecture Overview aims to support the architects during the architecting process. For that, the A3 Architecture Overview has to be tailored to the architecting process (see Section 4.2.4). In addition, the A3 Architecture Overview aims to meet architects needs (see Section 4.1.2) so the tool can be used by architects while architecting new products or product evolutions.

Encourage Knowledge Sharing

The A3 Architecture Overview focuses on practical aspects rather than technology capabilities or other complicate means to ensure easy deliver of knowledge to the stakeholders. It does not aim to use state-of-the-art technology but what works in practice. The A3 Architecture

Overview will not provide a solution to architecture knowledge sharing from a technology perspective to avoid dependencies to software tools.

Finally, to encourage the consolidation of knowledge, the A3 Architecture Overview aims to provide a simple an easy solution to the (apparently) hard task of consolidating implicit architecture knowledge (see Section 5.1.2), rather than to provide an automated or complicated solution.

Enable Effective Communication

The A3 Architecture Overview aims to provide a shared model that takes into account the human and organizational factors that produce architecture noise in order to avoid major communication barriers. For that, the A3 Architecture Overview aims to deliver the right information in a fashion that can be understood by a wide variety of stakeholders.

To support understanding, the A3 Architecture Overview will not force the use of architecture standards or modeling languages but will rather focus on providing visual representations and simple notations that are easy to understand.

8.2 A3 Architecture Overview Design

In order to provide a tool to achieve the previous objectives, we will design a tool to meet the requirements identified in Part I of this Thesis (see Section 6.3). In this section, those requirements are grouped and allocated to different aspects of the design, such as physical design, elements and visualization, structure, and inner design, and provide a design solution to meet them.

8.2.1 A3 Architecture Overview: Physical Design

In order to meet the tool objectives, as shown in Table 8.1, the physical design requirements that the tool should meet are presented.

Table 8.1: Physical Design Requirements

Physical Design Requirements
Require small overhead
Do not depend on custom-made software tools
Easy to use
Provide limited amount of information
Use an appropriate size to display complex information
Maintain a consistent way of communication
Improve existing written mechanisms
Encourage the dissemination of knowledge
Deliver the right information to the stakeholders while keeping the irrelevant part of information low
Support communication across disciplines and departments

To meet those physical design requirements based on our experiences with other tools (see Chapter 6), we chose a standard A3 paper size for the physical design of the tool, as shown in Figure 8.2. By designing the tool based on an A3 paper size, we meet our requirements by:

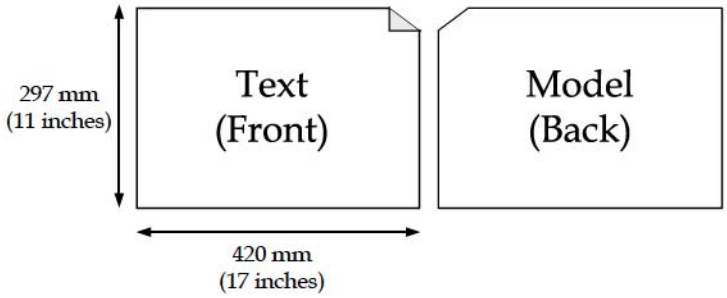


Figure 8.2: Communication Tool Physical Design

- We force a predefined paper size that limits the maximum amount of information that the tool can provide. Smaller sizes such as an A4 also meet the requirement, however they are usually too restrictive in the amount of information the tool can display. An A3 enables delivering more information without overloading the user (see Section 10.1.4).
- We provide an appropriate size to display complex information. An A3 fits well within the field of view, even when the field of view is reduced in the presence of complex information.
- We use a standard size, which will ensure the use of a consistent way of communication (always an A3 sheet of paper).
- We improve an existing written mechanism, by choosing an standard paper size to display information(although distribution can be electronic).
- We only require from the architect (or other A3 Architecture Overview creator) to create an A3, instead of a long document, so the overhead is expected to be at least smaller than writing a document.
- We provide an easy to use tool, that does not depend on software tools, that can be easily taken to meetings, etc.
- We encourage the dissemination of knowledge by requiring only to create/read one A3 rather than long documents or large diagrams.
- The tool does not depend on custom-made software tools as it is a paper-based approach.
- We support delivering the right information to the stakeholders while keeping the irrelevant part low, as the A3 forces brevity and synthesis of knowledge.
- We support communication by avoiding information overload as we provide just an A3.

As shown in Figure 8.2, the A3 Architecture Overview physical design separates text and model. Physical separation of text and model enables the A3 Architecture Overview to be used both individually and as a collaboration tool. For communication (e.g. at meetings) mostly visual representations are needed, as textual descriptions are usually not useful during

discussions (see Section 10.2). By allocating text and model in different sides of the same A3 sheet, users focus either on the model or the text. This way, the model can then be used at discussions without being disturbed by text. The text is important to complement the model with additional information and to enable individual use of the A3 Architecture Overview. In addition, it makes the A3 Architecture Overview an independent piece of information that does not need to be embedded anywhere else (e.g. in a document, see Section 6.1.1).

As shown in Figure 8.2, the physical orientation of the sheet is landscape. Landscape orientation is more convenient for human visualization as it fits better in the average field of view (this preference to display visual information in landscape orientation can also be observed in TVs, monitors, etc).

8.2.2 A3 Architecture Overview: Elements and Visualization

Once the physical design has been chosen, in order to meet the tool objectives, we gather the requirements that the elements and visualization provided by the tool should meet.

Table 8.2: Elements and Visualization Requirements

Elements Requirements
Provide limited amount of information
Provide a shared view
Deliver the right information to the stakeholders while keeping the irrelevant part of information low
Visualization Requirements
Use visual representations
Keep the notation simple
Limit the amount of visual attributes and ensures differences among them
Do not depend on custom-made software tools
Understood by a wide variety of stakeholders
Ensure that the information is conveyed and interpreted correctly
Prevent the lack of system overview
Support communication across disciplines and departments

Elements

To meet the tool objectives, the tool will focus on architectures in order to support product evolution. For that, the A3 will provide architecture knowledge. Therefore, the elements chosen to be part of the tool are the different types of information that belongs to the architecture knowledge (see Section 4.2.1). An appropriate visualization is chosen for each type of information. The format chosen to display that information is an architecture overview as described in Section 7.1.1. It is to say, the elements to be included in the A3 will be presented in the form of an architecture overview. This means that architecture knowledge will be presented by using three views; functional, physical and quantification, and by text and visual representations for the additional information.

By choosing an architecture overview as an architecture representation to capture architecture knowledge, we meet some of the above requirements by:

- We provide a limited amount of information by using a limited number of views to capture architecture knowledge.
- We provide a shared model (an architecture overview) that can be used by a wide variety of stakeholders instead of providing different views for different stakeholders.

- We provide the right information to the stakeholders while keeping the irrelevant part of information low by providing only essential information.

Visualization

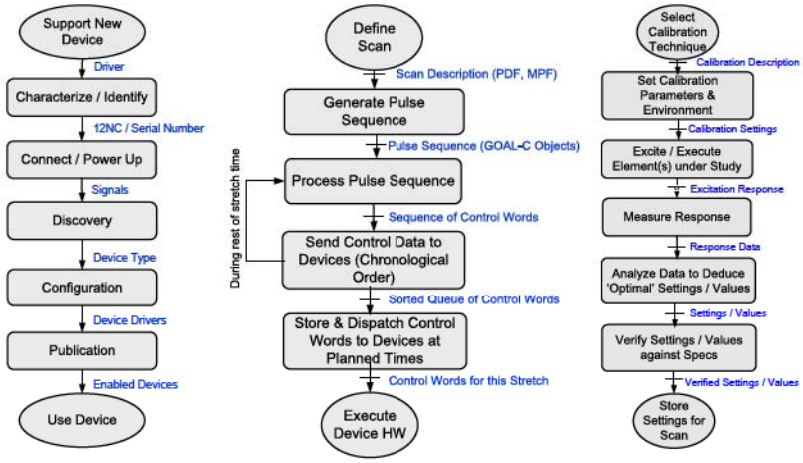
To visualize the architecture overview, we chose a convenient representation for each architecture knowledge element (see Section 4.2.1) that must be present in the architecture overview. To chose a convenient representation, we use the experiences obtained in different projects to know what works and what does not work in an industrial environment (see Chapter 6). Based on those, the architecture knowledge elements will be visualized as:

- **Functionality**, captured in the functional view of the architecture overview, will be visualized as a functional flow, as shown in the examples provided in Figure 8.3.
- **Architecture structure** (connectors and interfaces), captured in the physical view of the architecture overview, will be visualized as a building block view, as shown in the examples provided in Figure 8.4.
- **Quantification**, captured in the quantification view of the architecture overview, will be visualized either in textual form, as shown in Figure 8.5(a), or through tables as shown in Figure 8.5(b). Tables are a useful way to organize and communicate a complex set of ideas or data effectively.
- **Problem and solution** and **design decisions and rational**, captured in the additional information view of the architecture overview, will be visualized by using both text and visual aids, as shown in Figure 8.6.
- **Important stakeholder's concerns**, captured in the additional information view of the architecture overview, will be visualized by using a visual representation, in which system concerns are represented by keywords and classified using a 4-column view, as shown in Figure 8.7.

Those visualizations, in order to meet the tool objective of providing a system view, will be visualized from a system perspective. That in practice means, as that a top level system view will be used as a baseline to create the detailed views (see Section 9.3). By choosing those visualizations for the architecture knowledge elements, we meet the visualization requirements by:

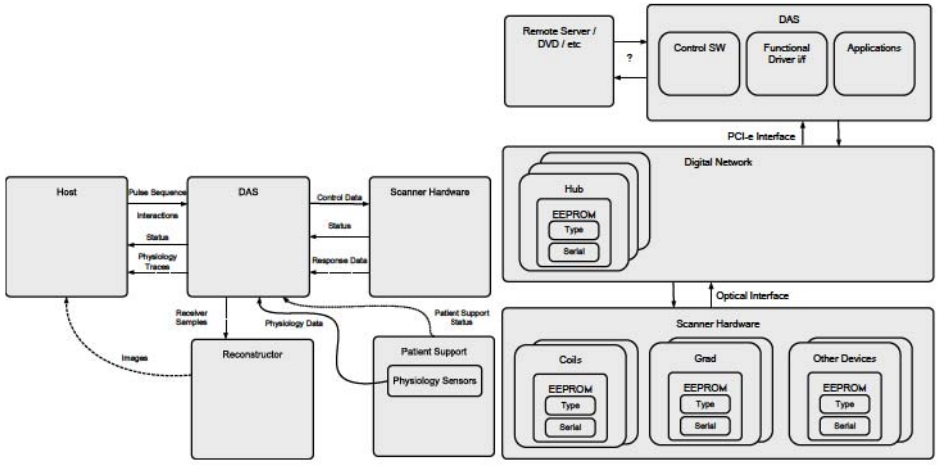
- We use visual representations to visualize architecture knowledge elements, in a simple and compact way that can (hopefully) be understood by a wide variety of stakeholders.
- We keep the notation simple, and the guidelines to create those visualizations (see Section 9.5) limit the amount of visual attributes and ensures differences among them.
- The tool does not depend on custom-made software tools or modeling languages to create the visualizations.
- We ensure (to some extent) that the information is conveyed and interpreted correctly by using visual representations, supported with textual explanations when needed.
- We provide a system overview by providing the essential information in the form of an architecture overview, in which the views are visualized form a system perspective.

- We support communication across disciplines and departments by providing a shared view that uses simple, easy to understand notation, instead of standards, domain languages, etc.



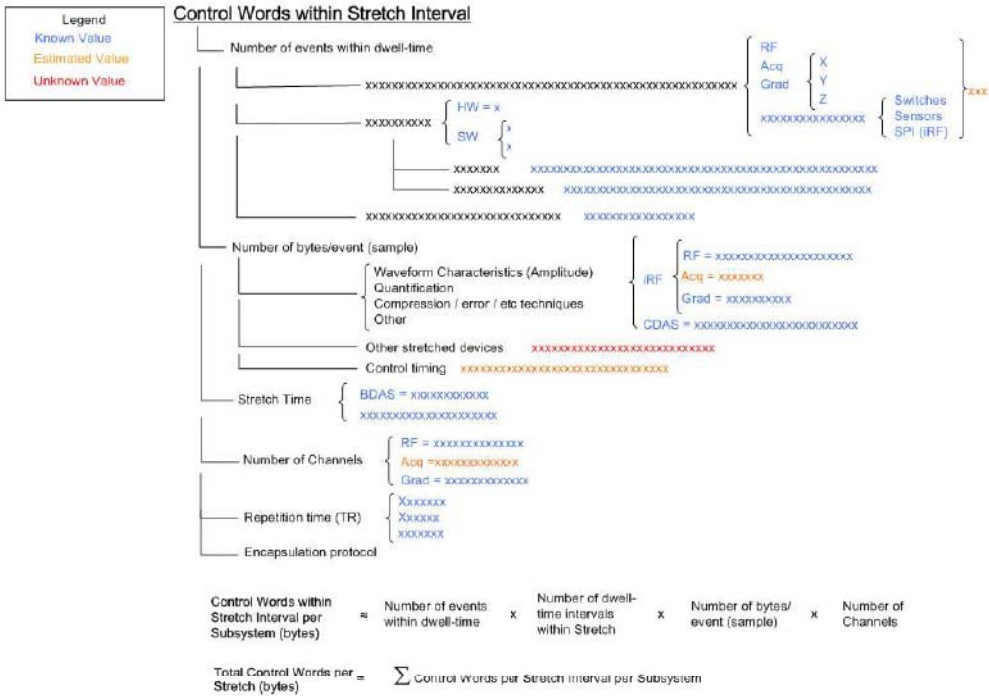
(a) Discovery of MRI Network Devices (b) Control of MRI Scanner (c) MRI System Calibrations

Figure 8.3: Functionality: Functional View Examples



(a) MRI Control Communication (b) MRI Discovery Components

Figure 8.4: Structure: Physical View Examples

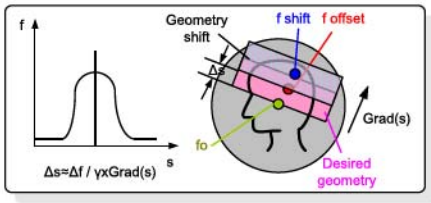


(a) Quantification of Key Parameters

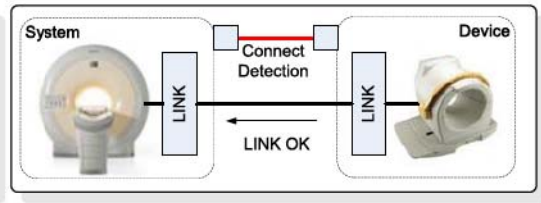
			Economy	Standard	Scaled Performance
Network		iRF Hubs	1	1 +1 Recon	2 +1 Recon
		Lanes / Links	3	4	8
		Bandwidth	X MB/s	X MB/s	X GB/s
Transmit		Links	1	1	2
		Bandwidth	X MB/s	X MB/s	X MB/s
		Channel Count	X channels @ 1 MHz	X channels @ 1 MHz	X channels @ 1 MHz
			X channels @ 2 MHz	X channels @ 2 MHz	X channels @ 2 MHz
Gradient		Links	1	1	1
		Bandwidth	X MB/s	X MB/s	X MB/s
		Channel Count	X channels @ 160 KHz	X channels @ 160 KHz	X channels @ 160 KHz
Receive		Links	1	1 (2)	2 (4)
		Bandwidth	X MB/s (in/out)	X MB/s (in/out)	X MB/s (in/out)
		Channel Count	X channels @ 1 MHz	X channels @ 1 MHz	X channels @ 1 MHz
			X channels @ 2 MHz	X channels @ 2 MHz	X channels @ 2 MHz

(b) Quantification of Key Parameters Through Tables

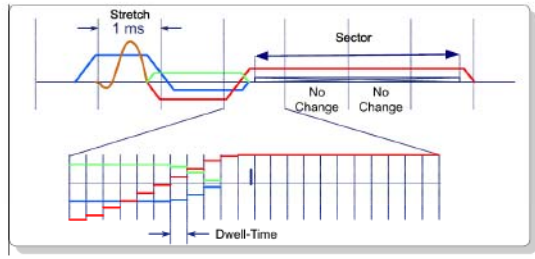
Figure 8.5: Quantification: Quantification View Examples



(a) Geometry Shift due to Frequency Offset



(b) Connection of MRI Devices



(c) Sampling of MRI Pulse Sequences

Figure 8.6: Visual Aid Examples

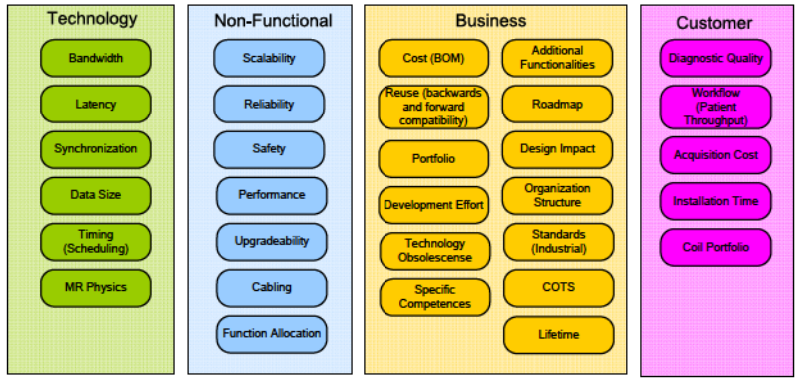


Figure 8.7: Important Stakeholder's Concerns: System Concerns View Example

8.2.3 A3 Architecture Overview: Structure

In previous sections we have chosen the physical design, the elements and how they are visualized. In order to meet the tool objectives, as shown in Table 8.3, we gather the requirements that the tool structure should meet.

The structure of the A3 Architecture Overview aims to ensure the integration of the individual elements in one physical place. With a good structure stakeholders can rapidly find the information that is relevant for them, and they can process that information more easily. Providing structure to an A3 improves readability and comprehension. For that, the allocation of views and text into a predefined structure will be based on the reader visual flow; from top to bottom, left to right[3], as shown in Figure 8.8.

[3]Western readers

Table 8.3: Structure Requirements

Structure Requirements
Maintain a consistent means of communication
Improve existing written mechanisms
Easy to use
Help finding the required system information

In the A3 Model, as shown in Figure 8.8, views will be arranged left to right from more stable to less stable over time. The functional view is more likely to remain unchanged over time, for that reason will be the backbone of the A3 Model and the first view the reader should look at. Then, visual aids are placed next to the functional view to support understanding and creating the correct mental model. Then quantification and physical views (quantification and physical view can exchange position in the structure), which are more likely to change over time are provided. Annotations are placed in the available white space left in the A3 Model (see Figure 8.8).

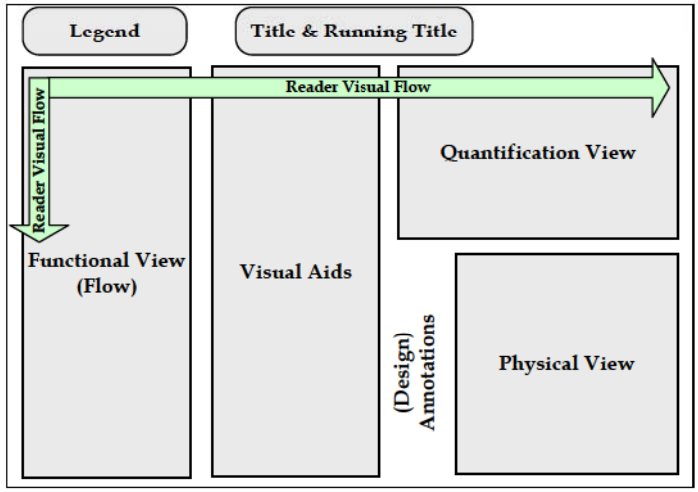


Figure 8.8: A3 Model Structure

In the A3 Summary, as shown in Figure 8.9, the different sections will be arranged in a way that support the reader to obtain a coherent story. Sections are boxed in the A3 Summary to enable the reader to locate the information easily and to structure the text.

By choosing the previous structure for the architecture overview, we meet the above requirements by:

- We improve existing written mechanism by supporting readability. By placing individual elements in a pattern that is easily recognizable, the structure helps in discerning and remembering which elements there are and which relationships are relevant to consider.
- We make it easy to use by supporting the processing of information. A clear pattern makes it easy for the eye to come back to objects that were already perceived, and thus supports processing [Koning, 2008].
- We keep a consistent way of communication by providing a predefined structure. The structure guarantees the consistent integration of the partial views from the architecture overview, and consistency across different architecture overviews.

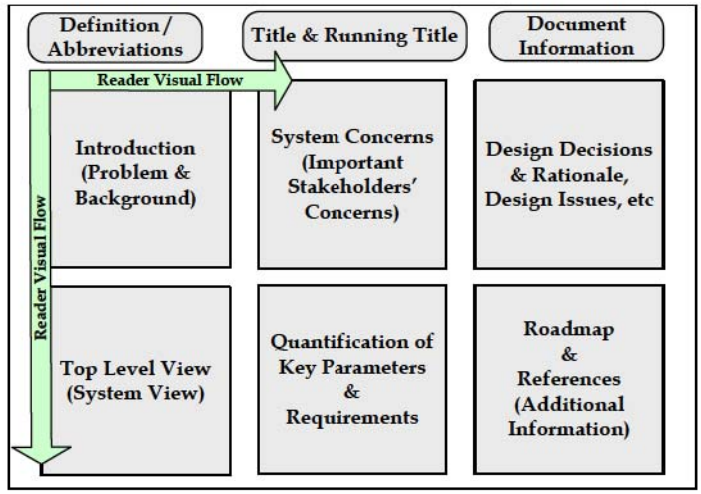


Figure 8.9: A3 Summary Structure

- We provide a common structure that helps the reader to identify easier whether or not the architecture overview is of any interest to him, and to quickly find where specific (system) information is.
- We enable finding the system information in an easy way by providing structuring in a consistent way (across A3 Architecture Overview) that enable readers to become familiar with the ordering principle of the architecture information contained. The structure also trains the reader into a specific patter, enabling finding the information easily in A3s with the same pattern.

8.2.4 A3 Architecture Overview: Inner Design

In order to meet the tool objectives, there are some additional aspects of the design that need to be incorporated into the tool. These requirements are shown in Table 8.4.

Table 8.4: Inner Design Requirements

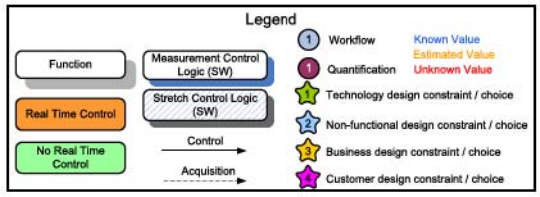
Inner Design Requirements
Keep the notation simple
Enable a flexible way to share knowledge
Appealing
Support the creativity of architects
Prevent the lack of system overview
Support to estimate the impact of change

Additional Elements

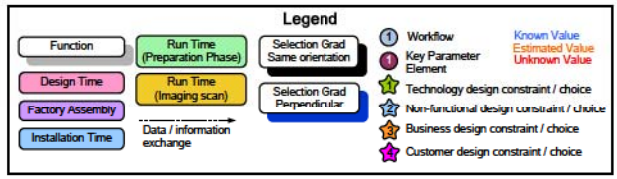
As shown in Figures 8.12, 8.13, some additional elements are needed to enable the tool to meet all the objectives; a **Legend**, **Document Information**, and **Definitions/Abbreviations**.

A Legend is needed to clarify the reader the notation used in a particular A3 Architecture Overview. As shown in Figure 8.10, a Legend describes the notation used in the specific A3

Model such as color coding, shapes, icons, and any other elements used in the views, in order to help the reader understand the information provided.



(a) Legend for A3 Model: MRI Resonance Frequency Calibration



(b) Legend for A3 Model: MRI Generation of Control Words

Figure 8.10: Legend Examples

The Document Information element is needed to identify the A3 Architecture Overview within the documentation system of a specific company. It provides the specific information that any document should have, such as author, document identifier, version, date, etc. Finally, the Definitions/Abbreviations element id needed to clarify any abbreviation or new concept that has been used in the A3 Summary, in order to support reader in understanding the text provided.

Relation among Views

To take advantage of having different views within the same A3 Model, those views are explicitly mapped to each other as much as possible. As represented by arrows in the Figure 8.12, views within the A3 Model are not isolated from each other. The specific mapping among views within the A3 Model are:

- Main functions allocated into physical elements, as shown in Figure 8.11.
- Quantification data mapped to other views, as shown in Figure 8.12.
- Visual aids mapped to functions, as shown in Figure 8.12.
- (Design) annotations mapped to physical, functional and/or quantification elements, as shown in Figure 8.12 (represented by star icons).

Links across views (e.g. by using numbers, icons, etc) enable to map one view to another. This linking method is chosen for the A3 Architecture Overview as dedicated connectors (e.g. lines) usually make the diagram look messy or overcrowded, and are not always easy to follow in a model.

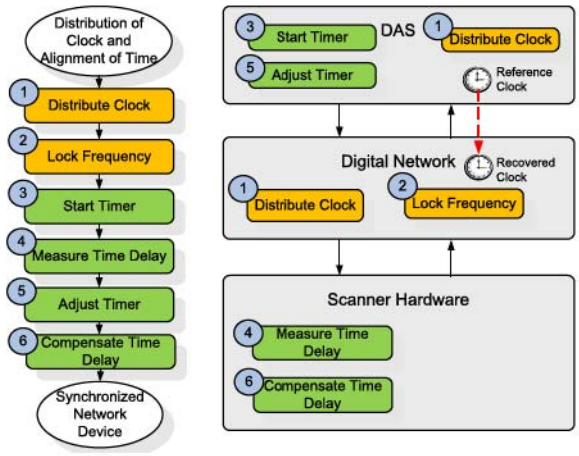


Figure 8.11: Mapping Functions onto Physical Elements

Flexibility

To incorporate flexibility in the tool design, the A3 Architecture Overview author can choose his preferred style to represent views by using his own notation, preferred font, color coding, icons, etc (see Figures 10.9, 10.12, 10.11, 10.12 for different styles).

Flexibility is also provided in the tool by allowing the A3 Architecture Overview author to change the size of the section boxes as well as by adding more sections when needed, etc. For example the box size in Figure 8.13 is variable. This means that if more room is needed for one element in the A3 Architecture Overview, that section can be expanded, or a new section created, as shown in Figure 8.13. However it should be noticed that as the A3 size is limited, resizing and adding more sections implies that there is less room for other sections.

By providing additional elements, relation among views, and flexibility to the A3 Architecture Overview we meet the additional requirements by:

- We support keeping the notation simple to understand by providing a legend that clarifies the notation to the reader.
- We enable a flexible way of representing architecture knowledge as different styles can be applied to the A3 Architecture Overview, and the elements allow different ways of visualization.
- We provide an appealing way to share architecture knowledge in the sense that it is compact, visual, and easy to use.
- We provide a system overview by encouraging that each view is represented from a system perspective, and by providing explicit relation among the views to support the overview.
- We support to better estimate the impact of change by providing explicit relation among different views, as well as a system perspective.

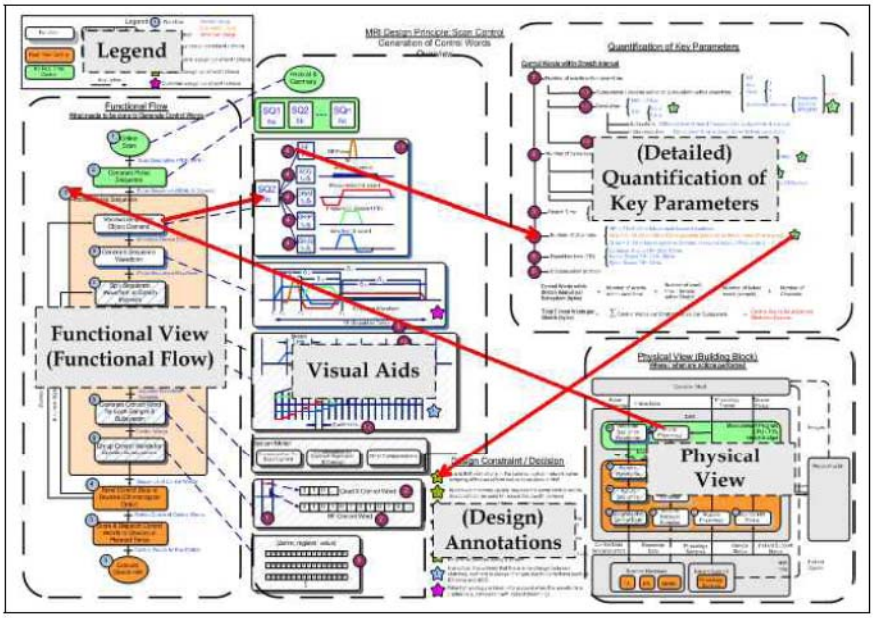


Figure 8.12: Inner design of an A3 Model

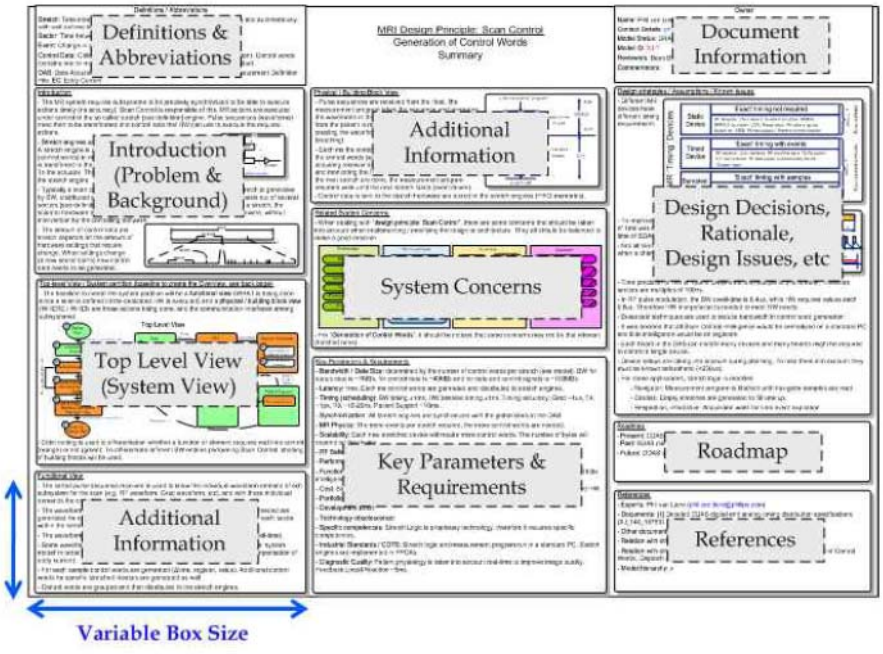
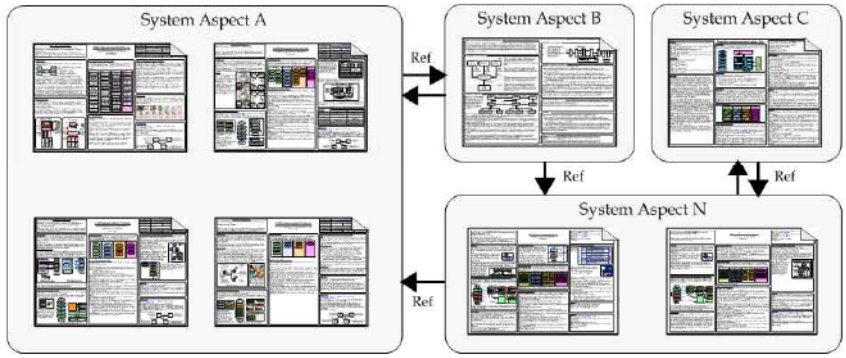


Figure 8.13: Inner design of an A3 Summary

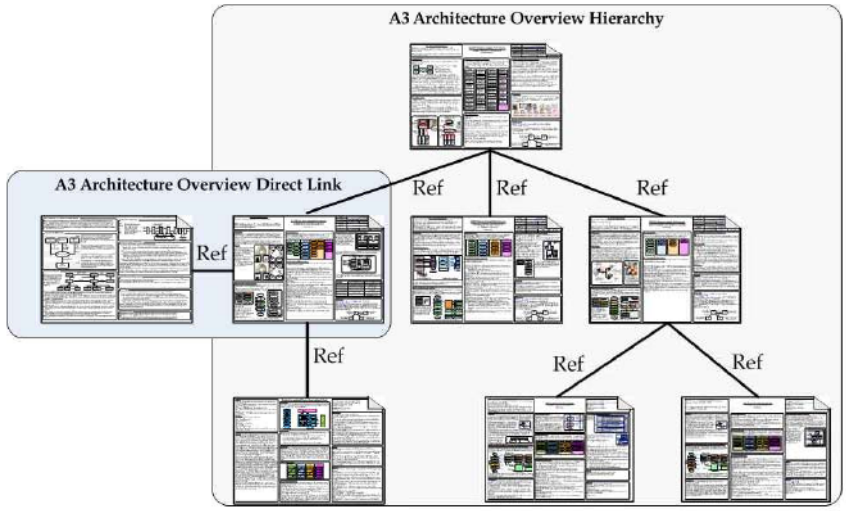
8.3 A3 Architecture Overviews as a Repository of Architecture Knowledge

A3 Architecture Overviews are linked to other A3 Architecture Overviews in several ways (described in the References section of the A3 Summary). As shown in Figure 8.14, there are three ways in which A3 Architecture Overviews can be linked;

- direct link to another A3 Architecture Overview as shown in Figure 8.14(a);
- hierarchy link to parent or child A3 Architecture Overviews as shown in Figure 8.14(a);
- and related system aspect link as shown in Figure 8.14(b).



(a) A3 Architecture Overview Hierarchy and Direct Link



(b) A3 Architecture Overview System Aspect Relation

Figure 8.14: A3 Architecture Overviews as a repository of architecture knowledge

Those links are likely to relate all A3 Architecture Overviews used collectively to describe a system. It is unlikely that an A3 Architecture Overview is not linked to another A3 Architecture Overview. A set of A3 Architecture Overviews forms a repository of architecture knowledge.

From the requirements identified in Part I of this Thesis (see Section 6.3), we observe that some of those are related to the repository of architecture knowledge.

Table 8.5: Architecture Knowledge Repository Requirements

Architecture Knowledge Repository Requirements
Record changes in the architecture knowledge repository
Enable reusing knowledge from previous experiences and products in current developments
Help keeping a structured overview of what has been communicated with a stakeholder

By having the architecture knowledge repository in the form of A3 Architecture Overviews, we meet the above requirements by:

- We enable to record changes in the architecture knowledge repository by identifying the A3 Architecture Overviews that are affected by a change.
- We enable the reuse of knowledge by having the architecture knowledge explicit in a manageable set.
- We help keeping a structured overview of what has been communicated with a stakeholder by knowing which A3 Architecture Overviews have been shared with the stakeholder.

8.4 Conclusions

An A3 Architecture Overview is a tool designed for knowledge sharing and effective communication of architecture knowledge. An A3 Architecture Overview provides a framework in which key architecture information obtained during the reverse architecting process is consolidated in order to share architecture knowledge.

The A3 Architecture Overview aims to support architects in their activities during product evolution. For that, the tool has to meet the following objectives; aid in product evolution, provide a system viewpoint, support the architecting process, encourage knowledge sharing, and enable effective communication.

To design the tool, requirements from previous chapters have been collected and allocated to different aspects of the design, such as the physical design, the elements and their visualization, the structure, and the inner design. For the physical design, an A3 paper size has been chosen. Both sides are used to display text and model respectively, in landscape orientation. The elements that the A3 paper display will be those related to an architecture overview. Those elements will be visualized with a functional flow, a building block view, quantification through text or tables, and text and visualizations for the additional information. For the structure, the reader's flow of view is used to allocate the different visualizations in a consistent pattern. For the inner design, a legend to clarify the notation is needed, explicit mapping among views and flexibility.

Finally, as A3 Architecture Overviews are linked to other A3 Architecture Overviews in different ways. A set of linked A3 Architecture Overviews forms a repository of architecture knowledge. By having architecture knowledge repository in the form of A3 Architecture Overviews, users can use A3 Architecture Overviews individually or collectively depending on their needs for architecture knowledge.

Chapter 9

Creation of A3 Architecture Overviews

This chapter provides a step-by-step guide to consolidate architecture knowledge in the form of A3 Architecture Overviews through the reverse architecting process. Skills desired to create readable A3 Architecture Overviews are discussed. Guidelines on form and style, self-evaluation checklists, and some guidance to avoid problems when creating A3 Architecture Overviews are provided. Finally, other A3 Architecture Overviews styles and common mistakes when creating A3 Architecture Overviews are discussed.

In previous chapters a **reverse architecting process** to consolidate implicit architecture knowledge (see Chapter 7), and the **A3 Architecture Overview** as a tool to capture that knowledge (see Chapter 8) have been introduced. This chapter brings those two together, providing a step-wise guide to consolidate architecture knowledge in A3 Architecture Overviews through the proposed reverse architecting process. To support the creation of A3 Architecture Overviews, guidelines on form and style are also provided. As an aid to self-evaluate the quality of an A3 Architecture Overview, criteria in the form of checklists are included. In addition, some guidance is provided to avoid common mistakes when creating A3 Architecture Overviews. Finally, other A3 Architecture Overviews styles and common mistakes when creating A3 Architecture Overviews are discussed.

9.1 Skills Required

Although the guide aims to support the architect in the creation of A3 Architecture Overviews, in practice, other stakeholders may also want to create an A3 Architecture Overview. Different stakeholders, however, have different ways of representing their knowledge (e.g. mathematical formulas), which may be difficult to understand by other stakeholders. Therefore, it should be taken into account that creating readable A3 Architecture Overviews may require some skills. To create good A3 Architecture Overviews the following skills are desired:

- **filtered reading and listening**, to be able to extract valuable and relevant pieces of information from the large amount available;
- **ability to describe and synthesize**, to be able to present the information in an easy to understand way;
- **ability to grasp new concepts quickly**, to be able to acquire knowledge from other domains besides the own domain; and
- **genuine interest for solving other people's problems**, to be willing to spend some effort to consolidate the gathered knowledge in an adequate way.

To support the development of those skills to create A3 Architecture Overviews, guidance in the creation process is needed. The following sections are aimed to provide support in the creation of readable A3 Architecture Overviews.

9.2 Identifying System Aspects

The first step in any communication process is to decide what to communicate (the message, see Section 5.1.1). Then, the next step is to put the message into context. For that, system aspects are used as the context for the knowledge to be communicated. System aspects may be system attributes, system characteristics, distinct features of the system, etc.

As shown in the following example (see Section 10.1 for the example's project description), once the message to be communicated is clear, it is necessary to identify the aspect that the message is dealing with, it is to say, what knowledge the message wants to communicate. Then, the knowledge that wants to be communicated must be placed in the system context. For that, questions like the ones provided in the following example can be used.

Message (Question): What do I want to communicate?
Aspect (Question): What should the reader learn?
System Aspect (Question): Where does this knowledge fit in the system context?

Message (Answer): How control words are distributed over the digital network of the MRI Scanner
Aspect (Answer): Distribution of control words
System Aspect (Answer): Control Scan

The system aspect is then used as the starting point for the A3 Architecture Overview. It will become the title of the A3 Architecture Overview, and the specific aspect that wants to be communicated will become the running title.

Allocating the message to be communicated into a system aspect is important to provide a system view rather than a localized view, and to provide a common framework in which other knowledge related to that system aspect can be incorporated. Once the system aspect to model is known, the step-by-step guide presented in the next section can be applied.

9.3 Step-wise Guide to A3 Architecture Overview Creation

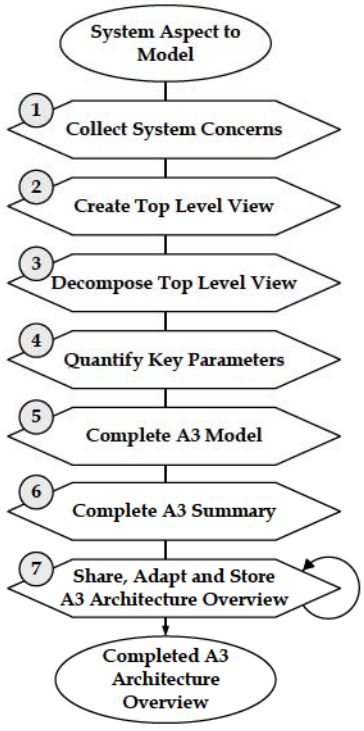
In this section we provide a step-by-step guide[1] to create A3 Architecture Overviews by following the reverse architecting process introduced in Chapter 7. The A3 Architecture Overview structure, as shown in Figure 9.1(b), will be used to capture and display the knowledge.

The goal of the step-wise guide is to provide a systematic process to capture the architecture knowledge gathered into different views (visual or textual), which will be mapped into a predefined A3 template. An example of A3 template is shown in Figure 9.1(b).

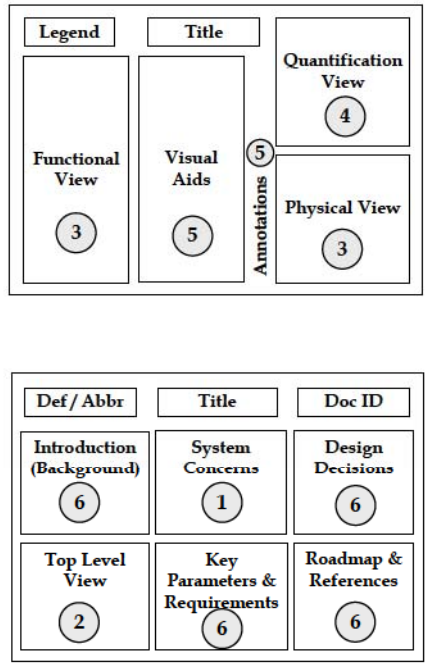
As shown in Figure 9.1(a), the creation process starts by selecting a specific system aspect to model. The goal of this process is to have key architecture knowledge of that system aspect captured in an A3 Architecture Overview. The reverse architecting process described in Chapter 7 will be used to consolidate the knowledge gathered in each step. For an efficient creation process, it is important to time-box each of the steps (e.g. 1-2 hours), to prevent spending too much time on any one step.

It should be noticed that completeness is not the goal. During the last step, the A3 Architecture Overview will evolve with the incorporation of insights and feedback obtained. The owner of the A3 Architecture Overview should decide when the knowledge contained in

[1]This guide has been consolidated in an A3 Architecture Overview CookBook, see Appendix D.



(a) Functional View



(b) Physical View (Top: A3 Model template; Bottom: A3 Summary template)

Figure 9.1: Step-wise Guide

it is enough, and consequently the A3 Architecture Overview is ready to be used (although good, still open to additional feedback). As a rule, once the steps have been completed, if feedback provided does not require relevant changes to the A3 Architecture Overview, it is ready to be used as a communication tool.

At the end of the process, an A3 Architecture Overview should be available to share the architecture knowledge and to communicate the message to all interested stakeholders.

1.- Collect System Concerns

As discussed in Section 4.2.1, anything that the stakeholders may think is important belongs to the architecture knowledge. Therefore, before diving into architecture representations, quantification, or other relevant aspects of an architecture, it is necessary to understand and capture what those important points are, that is to say, what are the system concerns of the architect and the relevant stakeholders. We apply the reverse architecting process to capture those system concerns:

- Information Extraction:

- **Stakeholder selection:** Select people who are responsible for the system aspect and its related elements, and those affected by changes in it.

- **Preparation:** From available information sources (e.g. documents) create a initial list of system concerns. Those system concerns may be present as keywords often addressed in those sources, words of warning, design issues, challenges, limitations, etc.
- **Questioning:** The initial questions to the stakeholders should aim to review the set of system concerns gathered during the preparation phase. Once feedback on those questions is obtained, other system concerns should be extracted by asking questions such as:

  What should we not forget when <designing / evolving> a solution to cope with <system aspect>?

  For <system aspect>, what <technological / functional / business / customer> concerns should be taken into account?

  What worries you about <system aspect>?

- **Update and validate:** Update the initial set of system concerns with the new information gathered. Distribute the list to key stakeholders to validate the list of system concerns.

- Abstraction:

- **Limit the amount of information:** Limit the number of system concerns to a manageable set (e.g. 10 - 15 system concerns).
- **Filter information:** Use keywords to describe a system concern (1-4 words). Remove or rephrase those concerns which are implementation specific (e.g. *"Ethernet does not provide enough bandwidth"* to *"communication bandwidth"*), and remove those which are likely to be solved in a near future.
- **Group information:** Merge similar system concerns. Label concerns according to a predefined category: *Technology, functional, business, customer*. If a concern seems to fit in more than one category, choose one.
- **Provide visual representations:** You may use bullet lists, tables or other ways to visualize the list of concerns. Instead of bullet list, tables, etc, we use a 4-column view (see Figure 8.7) to display system concerns. Color coding aims to differentiate the classification of system concerns according to their category.

- Presentation:

- **Select appropriate architecture view:** System concerns belong to the additional information view from the architecture overview. There is not a specific architecture view in the A3 Model. Text and visual representations will be used to capture and display that information.
- **Complete information:** Clarify whether a system concern is treated in some other place or A3 Architecture Overview and provide references. Clarify which of those system concerns will be specifically dealt with in this A3 Architecture Overview.
- **Place outcome in the architecture overview template:** The system concerns view belongs to the A3 Summary. The visual representation as well as additional textual description will be placed in the system concern section (box) of the A3 Summary (See Figure 9.1(b)).

At the end of this step, we should have a clear representation of key system concerns related to the system aspect under study. Most likely not all those concerns will be addressed within this A3 Architecture Overview, however it is important to include them so the reader is aware that there are more things to be considered about this system aspect than what is described in this particular A3 Architecture Overview.

2.- Create Top Level View

The next step in the process is to create an appropriate system partition or top level view that will be used as a starting point for the reader and for decomposing our system. A system view should be provided (see Section 8.1), even when the system aspect seems localized. This is necessary to help provide overview and to understand how parts fits in the whole. An example is shown in Figure 9.2.

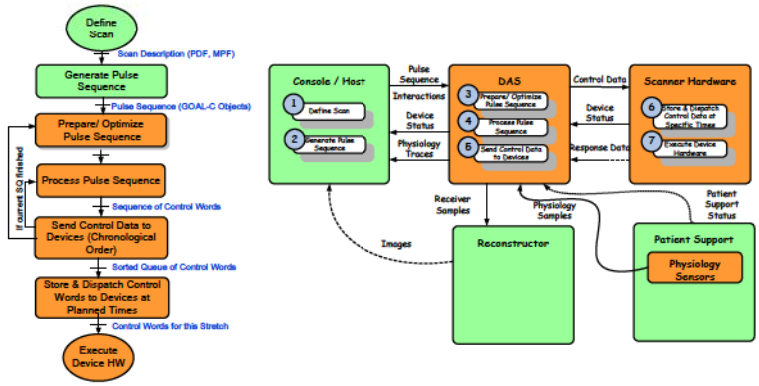


Figure 9.2: Top Level View Example

- Information Extraction:

- **Stakeholder selection:** Select those stakeholders that can put the system aspect into a context (e.g. fellow system architects). Selected stakeholders should have a view of the system as a whole.

- **Preparation:** Review existing system partitions used in architecture documentation or architecture representations to see whether the partitioning is consistent with those. Identify major building blocks and main functions related to this system aspect.

- **Questioning:** Physical building blocks and major functions need to be identified, as well as key interfaces and the allocation of the main functions into the building blocks. For that, questions to the stakeholders should be like:

Which major building blocks are involved in <system aspect>?

Which are the main functions that are involved in <system aspect>?

Which are the interfaces among building block needed for <system aspect>?

In which building block(s) are those functions performed?

- **Update and validate:** Stakeholders should validate the top level view created. An accepted top level view is necessary to keep consistency among different decomposition views that may be created from this top level view (see Figure 9.2) to highlight different viewpoints of the same system aspect.

- Abstraction:

- **Limit the amount of information:** As we only want a top level view, the view should be small. No more than 5-8 building blocks and 5-8 functions should be part of the top level view.

- **Filter information:** Only major building blocks and main functions should be kept. Only names (or identifiers) for the building blocks are required. Functions are represented in the form of *verb+noun* (e.g. send data).

- **Group information:** Allocate main functions to the major building blocks.

- **Provide visual representations:** Use blocks and arrows to represent the building blocks, functions and interfaces identified. Make a clear difference between blocks representing functions and blocks representing building blocks (e.g. by using different shapes, shading, etc). Arrange the functions in a functional flow in logical order. Finally, interfaces should be made explicit in the physical view, and functions should be allocated to building blocks.

- Presentation:

- **Select appropriate architecture view:** The top level view belongs to the additional information view of the architecture overview. There is not a specific architecture view in the A3 Model. Text and visual representations will be used to capture and display that information.

- **Complete information:** When describing the top level view, there may be different aspects that may be interesting to highlight. For that, visual strategies such as color coding, as shown in Figure 9.2, can be used. The rationale for the selected system partition must be provided as well.

- **Place outcome in the architecture overview template:** The outcome belongs to the A3 Summary. The visual representation as well as the textual description will be placed in the system partition box of the A3 Summary (See Figure 9.1(b)).

At the end of this step we should have a representation of the system in which physical building blocks and main functions related to the system aspect under study are present. This top level view will be used as a baseline for a more detailed decomposition in the following steps.

3.- Decompose Top Level View

The goal of this step is to decompose our system in a way that enables understanding the specifics of the system aspect under study. For that, the functional and the physical views from the top level view will be used for a detailed decomposition.

The purpose of the decomposition of the functional view is to highlight the WHAT; that is to say, what actions are required to achieve the system goal related to the aspect under study. The purpose of the decomposition of the physical view is to highlight WHERE and HOW these actions are implemented. This view is closest to the actual system.

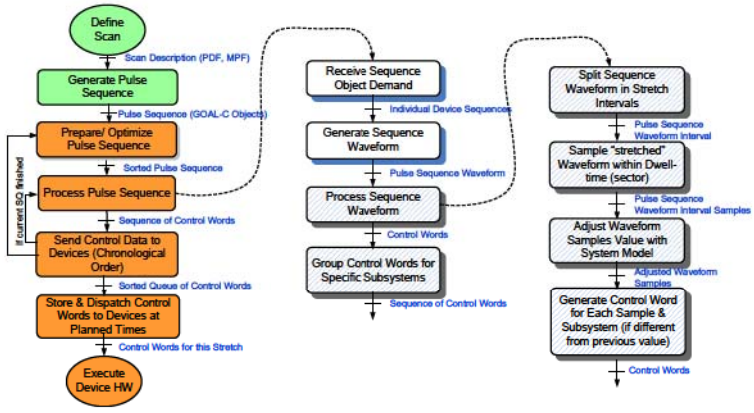


Figure 9.3: Functional Decomposition Example

- Information Extraction:

- **Stakeholder selection:** Experts with deep knowledge of the details and those who are related or affected by the system aspect under study.
- **Preparation:** Review existing documentation and related sources of information looking for information about the system aspect. Identify specific functions related to this system aspect and in which detailed building blocks those functions are performed. The top level view is used as a baseline to decompose and create a more detailed physical and functional view, as shown in Figures 9.3, 9.4.
- **Questioning:** The initial questions should aim to review the detailed views created during preparation phase. Once feedback on those is obtained, other system concerns should be extracted by asking questions such as:

  What actions are required to achieve <main function>?

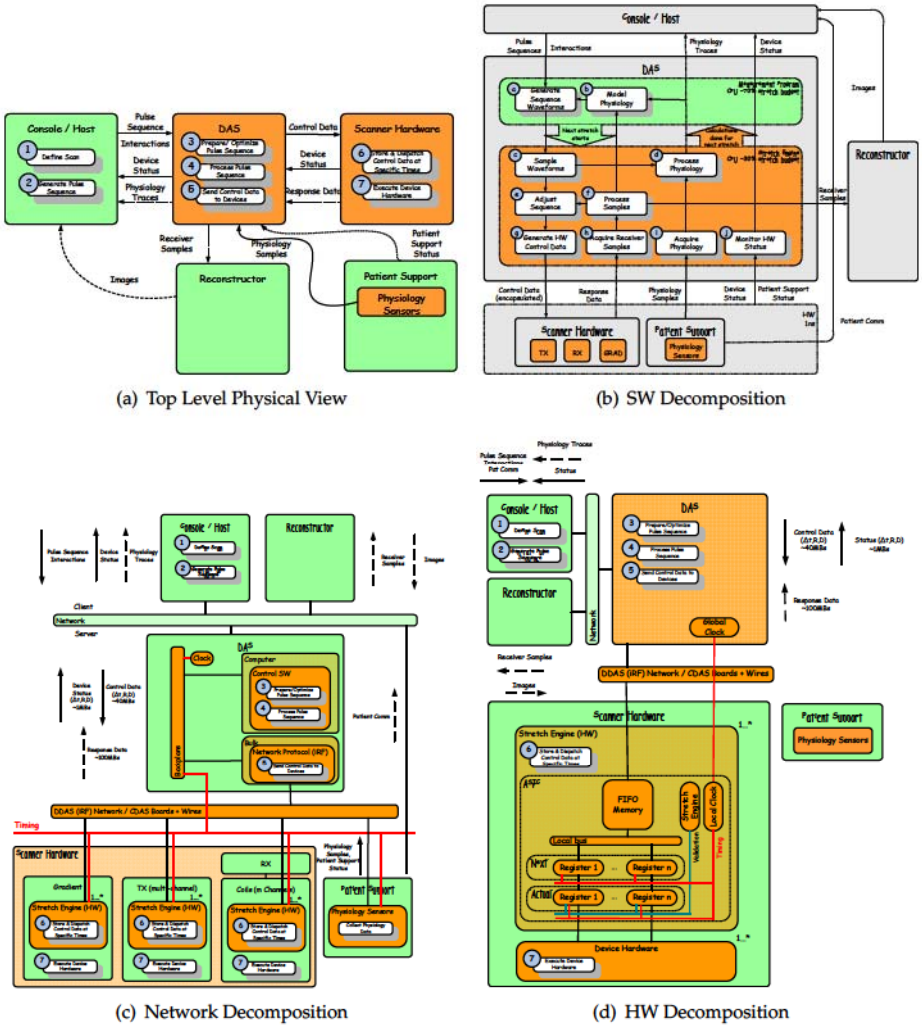
  How does <function> transform <input>?

  What physical elements from <main building block> perform <function>?

- **Update and validate:** New findings should be incorporated to the detailed views. The views should be validated against existing representations (if any) to ensure consistency.

- Abstraction:

- **Limit the amount of information:** As shown in Figure 9.3, no more than 5-8 functions per level should be used, and no more than two levels deep. In the physical view, no more than 5-8 physical elements should be part of the detailed view.



(a) Top Level Physical View

(b) SW Decomposition

(c) Network Decomposition

(d) HW Decomposition

Figure 9.4: Physical Decomposition Examples

- **Filter information:** Physical detailed elements that are important but not directly related to the system aspect under study should be removed.

- **Group information:** All major elements from the top level view should be part of the detailed views (see Figure 9.4). Then, assign detailed physical elements to a parent building block from the top level view and group detailed functions to their parent function. To describe elements, use general terms rather than specifics.

- **Provide visual representations:** To support the functional view, and to support the creation of the correct mental model, visual aids are desired. Visual aids, such as those presented in Figure 8.6, should be created to support the functional view.

- Presentation:

- **Select appropriate architecture view:** The detailed functional and physical view are part of the architecture overview. Due to the limited space in the A3 template, functions and grouped functions should be arranged in a functional flow (see examples in Figures 10.1, 10.2, 10.3, 10.4). The length of the functional flow should be less than the height of the A3 (297mm or 11 inches in landscape orientation) in order to fit in the template. The physical view building block view) should not be larger than half the heigth of an A3, which is the room allowed for the physical view in the A3 template.

- **Complete information:** Add some specific comments if needed to clarify some design decisions, choices made, etc. During the decomposition process those annotations should be made explicit.

- **Place outcome in the architecture overview template:** The physical and functional views belongs to the A3 Model. Visual aids should be placed next to the functional view. Annotations, such as those presented in Figure 8.12, should be placed in the white space available in the A3 Model (See Figure 9.1(b))).

At the end of this step, a major part of the A3 Model should be ready. Several views and additional information should already be present in the A3 Architecture Overview.

4.- Quantify Key Parameters

Although system aspects such as "flexibility" or "safety" may seem hard to quantify and even harder to measure there are some ways to increase understanding and ways to provide numbers to support those system aspects.

- Information Extraction:

- **Stakeholder selection:** Select those experts who can provide numbers (e.g. measurements) related to the key parameters to quantify.

- **Preparation:** The first step to quantify a system aspect is to find out what key parameters are needed to provide valuable quantification about that system aspect (e.g. key performance parameters). Once key parameters have been identified, relevant values such as current measurements and expected future values should be found, as well as ranges in which those parameters are used.

- **Questioning:** After reviewing the numbers obtained during preparation, estimations or expected values should be obtained for those parameters whose numbers cannot be found.

  Which is the typical/best/worst value for <key parameter>? Which is the future expected value for <key parameter>?

  While experienced practitioners may feel uneasy providing numbers, they can easily pinpoint a wrong estimation. Therefore, a questioning strategy to obtain estimations may be to provoke the expert:

Q: How much <power> do we need in <subsystem>?
R: I am not sure, I can't tell.
Q: Then should I put <500W> (absurd)?
R: No, that's way too much!
Q: So, should I punt then <50W>?
R: <40W> is more likely.

When using this approach, the confidence level of the values obtained should be annotated (e.g. high confidence for a measurement, low confidence for a wild guess).

- **Update and validate:** For the validation of the numbers it would be desirable to test the numbers against actual measurements to the current system. If that is not possible, credibility levels should be added to the numbers obtained.

- Abstraction:

- **Limit the amount of information:** At most 1-3 key parameters can be described in detail (see Figure 8.5). The rest of key parameters should just provide main values.
- **Filter information:** Choose measurements over estimations or guesses. Keep values with similar order of magnitude, that is to say, keep values that contribute to the key parameter and leave out small contributions (e.g. if the order of magnitude is the hour for the key parameter, do not add elements that contribute at the nanosecond order).
- **Group information:** Find how individual contributions add up, so a figure of merit can be created (see Figure 8.5). Easy to calculate approximations work better for an architecture overview than exact mathematical approaches.
- **Provide visual representations:** Graphics, charts and similar diagrams may be used to help visualize relevant values.

- Presentation:

- **Select appropriate architecture view:** An example of a quantification view is shown in Figure 8.5. Another way to present quantification data is through tables. Tables, as shown in Figure 8.5(b) are a useful way to organize and communicate a complex set of ideas or data effectively.
- **Complete information:** Provide credibility estimations for the numbers provided (e.g. by using colors to indicate whether the number provided is a measurement, an estimation, or a wild guess, see Figure 8.5(a)[2]). If a *requirement* is imposed to a key parameter, it should be made explicit.
- **Place outcome in the architecture overview template:** The detailed quantification view of key parameters belongs to the A3 Model (see Figure 9.1(b)). Other key parameters, their values and requirements (if any) belong to the A3 Summary.

At the end of this step, the major views from the architecture overview should be present in the A3 template.

[2]This approach will not work for color-blind people. Other approach may be to use e.g. bold for a measurement, normal for an estimate, and italic for a guess.

5.- Complete the A3 Model

Once all the individual views are placed in the correct place of the A3 template, to complete the A3 Model the following tasks are required:

- Make explicit links among individual views.
- Allocate annotations made during the process to a convenient location (available white space).
- Include a legend with the notation used in this A3 Model.

Once those task are completed, some time should be spent to fine tune the A3 Model to make it readable, appealing and to give the impression of completeness [Borchers *et al.*, 1996]. The guidelines presented in Section 9.5 can be used for this purpose.

6.- Complete the A3 Summary

Once the A3 Model is completed, the A3 Summary needs to be completed (see Figure 9.1(b)). For that, the following tasks are required:

- Add the document identification convention used at the company, as well as other relevant information such as owner, related product, date, version, etc.
- Develop the introduction. Provide background information to understand the system aspect and its importance, as well as context information. Place this in the Introduction section box (see Figure 9.1(b))
- Include design decisions, design issues and their rationale. Provide those aspects of the design that need an explanation, as well as why some decisions were taken and which problems or limitations the current design has.
- Create a roadmap. Explain how was the system aspect dealt with in the past, how is dealt with today, and whether some changes are expected in the future (see Figure 10.4 for an example of the MRI roadmap, based on Figure 2.5). Place this in the Design Decision section (box) (see Figure 9.1(b))
- Provide references. Include contact details of experts that can provide additional information, documents or other sources of information used, system aspects that are directly related to the one discussed in this A3 Architecture Overview, and other A3 Architecture Overviews that deal with other specifics of this system aspect. Place this in the Roadmap&References section box (see Figure 9.1(b))
- Complete other sections [optional]. More sections can be added depending on whether something important needs to be included and whether it fits on the current template (see Figures 10.1, 10.2). Create section boxes for those other sections and place them in the most convenient location in the A3 Summary.

As shown in Figure 8.13, each section of the A3 Summary is enclosed in a box, and has a predefined order designed to aid the reader to find the specific information he is looking for. Section order should help the reader to get a coherent story about the system aspect under discussion.

7.- Share, Adapt and Store

As the A3 Architecture Overview is drafted, the author should collect feedback and then integrate relevant feedback into the A3 Architecture Overview. The input is important not just to increase the quality of the A3 Architecture Overview but for considering as many viewpoints as possible and building alignment of thinking among stakeholders.

To that end, the A3 Architecture Overview should be shared among stakeholders, mainly those involved in the previous steps, in order to collect feedback and remove inconsistencies. This reviewing process should be repeated as many times as necessary, until only minor changes in the A3 Architecture Overview are required (see Section 9.4). Probably there will always be some remarks or comments about the A3 Architecture Overview. It is the responsibility of the author to decide whether proposed changes do or do not add value to the A3 Architecture Overview.

For distribution and future use of architecture knowledge, the A3 Architecture Overview should be stored in a known repository[3] or the place in which the company stores documentation. This is necessary in order to make the A3 Architecture Overview available to all employees so it can be used like any other document. If the A3 Architecture Overview is shared without being present in the repository (e.g. through email), there is the risk that the architecture knowledge will be lost or just available to a limited number of employees, affecting knowledge sharing.

9.4 *Reviewing A3 Architecture Overviews*

Finding people capable of critical review should not be difficult in a large company. For individuals, in this section a checklist of the most common questions asked during A3 Architecture Overview reviews is provided in Table 9.1. Authors can use these questions in self-critique and in anticipation of a review.

Toyota's A3 reports are always written and submitted to someone, such as a manager, who critically evaluates the report [Sobek II and Smalley, 2008]. In other companies such as Philips, where consensus among stakeholders is desired, A3 Architecture Overviews are reviewed by experts related to the system aspect, as well as by at least a system architect.

The first concern when creating an A3 Architecture Overview, is how to interest the targeted people of this knowledge. Nowadays most people are overloaded with information. If it does not look nice, it is easily put aside. For that reason, a key question to evaluate the A3 Architecture Overview should be:

Does the first impression stimulate the reader to dive into A3 Architecture Overview and take in the information contained in it?

The effectiveness of writing A3 Architecture Overviews increases greatly with practice, however, there is not a single way to implement A3 Architecture Overviews. When reviewing A3 Architecture Overviews, it should be taken into account that every implementation of an A3 Architecture Overview is unique, and reflects the experience, background and personal preferences of the author.

[3] This is not the case in the Toyota approach, in which each individual keeps his own A3 Reports. In the A3 Architecture Overview however the goal is to share knowledge, for that A3 Architecture Overviews have to be available and easy to access to the different stakeholders

Table 9.1: Evaluation Questions for A3 Architecture Overviews

Self-evaluation Questions
General
Is the system aspect clear and logically depicted?
If the focus clear and reflected in the contents?
Are all architecture knowledge elements described in this A3 Architecture Overview?
Is the A3 Architecture Overview easy to understand by a wide variety of stakeholders?
A3 Model
Are all relevant functions identified?
Is the physical decomposition clear?
Are key parameters quantified?
Are values for the key parameters provided and their credibility clear?
Are there enough visual aids to support the A3 Model?
A3 Summary
Is the textual information relevant?
Is it a repetition of what's already in the A3 Model?
Are there enough references provided?
Is the A3 Summary clearly structured?

9.5 Guidelines on Form and Style

For the creation of readable A3 Architecture Overviews, in this section we provide guidelines on form and style. Those guidelines are based on the experiences applying the approach in real projects, as well as from the feedback collected from experts using the tool. These guidelines are meant to assist architects or anyone interested in creating an A3 Architecture Overview.

A3 Architecture Overviews should be clear, neat, well organized and aesthetically pleasing. The form of an A3 Architecture Overview should aid the reader finding the information rather than hindering the communication of the contents. In practice this means; making ample use of white space, achieving good use of symmetry, lining up box edges, aligning headings, paragraphs, etc [Sobek II and Smalley, 2008]. As readability is paramount, the same text styles and size should be used in the whole A3 Architecture Overview, and the font size should be big enough for comfortable reading. In addition, uncommon terminology and jargon should be avoided.

The proportion of the different elements of the A3 Architecture Overview should be kept (e.g. no awkward sizes for the boxes in the A3 Summary). When placing boxes in the A3 Summary, the reader flow; from top to bottom, left to right, should be taken into account. The shapes used should be clear and their number limited. Clear distinction should be made between physical and functional elements in the A3 Model.

Color is a strong visual signal (for people who are not color blinded), so it is worth paying attention to it. Using a distinct color in a diagram for an object with a particular attribute, programs the meaning of that color for the rest of the A3 Architecture Overview. Although color can enlarge the appeal of the model, its use should be scarce, as it can overload the reader. Too much color may confuse the reader to make the model look "less serious". If assistance in the use of colors is needed, in [Chijiiwa, 1987] many examples of color combinations that can be used are provided.

The most salient characteristic of a good A3 Architecture Overview is its brevity. To achieve that, there should be no repetition, superfluous information or extra wording. The

A3 Summary should not repeat the information provided in the A3 Model. As the targeted users of A3 Architecture Overviews are diverse, the use of jargon and unfamiliar terminology should be avoided. Key points should be italic or bold, without overusing it.

Table 9.2: Guidelines for A3 Models

Guidelines for A3 Models
Readability
Use a font size big enough for comfortable reading (>10 ppt)
Do not use different font sizes (except headings) or styles
Provide enough white space (do not fill in information just for the sake of it)
Place title and running title in the top center of the A3
Layout
Keep the proportions of the A3 elements and boxes as equal as possible, even when one view seems more important than other
Take into account the "reader flow" from left to right, and from top to bottom
Place annotations in the most convenient location
Objects
Be clear about what different shapes mean
Do not use more than five shapes within the same A3 Model
Avoid making objects smaller to have more room in one view, or making objects bigger to fill up a view that otherwise looks too empty
Make a clear distinction between functions and physical elements (e.g. by using different shades for functions)
Colors
Do not use more than five colors within the same A3 Model
Be careful with color clutter (e.g. using the same color as used in other views such as System Concerns view)
Avoid colorblindness or problems when printing in black and white by using colors with different levels of brightness/lightness
Choose colors in the following order: blue, red, green, purple, orange and yellow [Koning *et al.*, 2002]
Avoid if possible the use of pink color
Visual Aids
Box visual aids to prevent overlapping
Label visual aids properly
Use tables to organize a complex set of data or ideas
Align of visual aids to the left

Table 9.3: Guidelines for A3 Summaries

Guidelines for A3 Summaries
Structure
Box every section clearly
Do not make a section box too large or with a awkward size
Keep order of sections across different A3 Architecture Overviews
Contents
Try to use pictures or drawings instead of large text descriptions
Do not try to fill in the A3 Summary just for the sake of it. Leave white space if possible
Use underlining and bold with caution
Style
Use a font size big enough for comfortable reading (>10 ppt)
Do not use different font sizes (but in headings) or styles
Text should be concise, there should be no repetition, superfluous information or extra wording
Do not use unfamiliar terminology or jargon
Keep company documentation style. The style of the A3 Summary should resemble company documents as much as possible
Be clear about the ownership of the A3 Architecture Overview and provide contact details
Treat every A3 Architecture Overview as an individual and independent document (e.g. document identification)

In the A3 Summary, each section should be clearly labeled, arranged in logical flow, and separated from other sections by being enclosed in a box, with enough margins between boxes. The title of the report, and the running title should be clear and in the center of the report. The title should unambiguously indicate the system aspect under study, while the running title should be clear about the specific focus of this A3 Architecture Overview. The author name, date, status and other short of information should be clear and located in the top right of the A3 Summary.

9.6 Other A3 Architecture Overview Styles

Changing the structure or changing the style does not mean changing the tool. Every person has his own style of working and thinking, and the A3 Architecture Overview tool enables enough flexibility to allow people to tune the A3 Architecture Overview to their preferred style. What is important is the underlying philosophy (see Section 8.1). Some examples of different A3 Architecture Overviews created by people with different backgrounds are shown in Section 10.4. In those we can observe that despite the differences in form and style such as the use of 3D models in the physical view, the use of black and white, different A3 Summary structure, etc. Despite those differences, all of these can be easily recognized as A3 Architecture Overviews.

9.7 Common Mistakes

During the creation of A3 Architecture Overviews some minor issues may arise. Those issues can probably be solved in many smart ways. In this section we do not aim to provide an extensive list of problems and their best solutions, but to describe some situations that may arise and ways to cope with them that has proven successful in real situations.

- **Trying to fit existing models:** It is tempting to reuse existing diagrams or models and try to make them fit into the A3. This in practice, leads to many readability problems and requires more time than creating a new view from the existing model. Instead of spending time trying to resize, remove, adapt and tune those views, time could be use in extracting the essential information from those views and creating new ones.

- **Neglecting the importance of one view:** We have observed that some people focus on one view (e.g. functional view), and spend plenty of time in that one. As a consequence, they have a great view, but little time is spent in the other views, resulting in poor views. This lead to an unbalanced A3 Architecture Overview which looks unfinished or incomplete. All views are important, therefore the time allocated to views should be time-boxed so proper views are created. During iteration, additional time can be spent on those views that require additional effort.

- **"Not everything fits":** In occasions, the amount of information collected is large and it has not been abstracted enough. This usually leads to fitting problems when placing the outcome in the A3. The common reaction is to create additional room in the A3 by reducing the font size, resizing the views, etc. This in the end lead to readability problems. The A3 Architecture Overview is not meant to contain lots of information, but essential one. If not everything fits, the reverse architecting process should be done again in order to find out what can be left out, how to better abstract current information, and/or a more effective way to present the information.

- **Problems creating a functional view for abstract system aspects:** Some system aspects such as "scalability" or "safety" seem not to have a clear flow. Many unrelated things may be done to cope with that system aspect that seem to have little relation, and of course no order or flow among them. However, from the communication perspective, when someone has to explain (in text or verbal communication) those concepts, usually a set of steps are used to describe how those abstract system aspects are dealt with. Verbal communication does not allow explaining multiple things at the same time; an order is required. That order is the one that should be used in the functional flow.

9.8 Conclusions

The creation of A3 Architecture Overviews is meant to support the architect in his duties. In practice, other stakeholders may also be interested in the creation of A3 Architecture Overviews. For that, is should be taken into account that creating readable A3 Architecture Overviews some skills are desired, such as filtered reading and listening, ability to describe and synthesize, ability to grasp new concepts quickly, and genuine interest for solving other people's problems.

The step-wise guide can be used to create an A3 Architecture Overview to consolidate the architecture knowledge related to a system aspect. The goal of the guide is to provide a systematic process to consolidate the knowledge into different views that will be mapped onto a predefined A3 template. The process consists of seven steps that use the reverse architecting process to collect, abstract and present the architecture knowledge in a convenient format for the A3 Architecture Overview.

As the A3 Architecture Overview is being drafted, the author should collect feedback to increase the quality and to remove inconsistencies. Finding people to review an A3 Architecture Overview should not be difficult in a large company. For individuals, a checklist with the most common questions asked at reviews is provided to self-critique and to anticipate for a review.

Guidelines in form and style are provided to contribute to the creation of readable A3 Architecture Overviews. Form and style can be as important as the content; an unappealing A3 Architecture Overview can be put aside.

To change the structure or the style does not mean changing the tool. Every person has a different way of working and the A3 Architecture Overview enables enough flexibility to tune the tool with the preferred style or structure.

Finally, during the creation of A3 Architecture Overviews, minor issues can arise. To avoid common pitfalls, such as trying to fit existing models or neglecting the importance of one view, we provide some recommendations to avoid those.

III Validation and Evaluation

Chapter 10

Application of A3 Architecture Overviews in Industry

In this chapter, experiences of applying A3 Architecture Overviews in industry are presented. Study cases in which the A3 Architecture Overview was used in real projects are described. The outcome and lessons learned from those experiences, and a comparison of A3 Architecture Overviews with text documents are provided. Finally, other case studies in which the A3 Architecture Overview was applied are used to evaluate the creation effort.

In Part I of this Thesis, popular approaches to deal with evolution of complex systems, approaches to share architecture knowledge and some methods to provide effective communication were presented and tested in real projects at Philips Healthcare MRI. In Part II, a reverse architecting process and a tool to consolidate architecture knowledge, in a way to support effective communication, were designed. In this Part III, we present the experiences of applying the A3 Architecture Overview in industry, in order to evaluate the applicability of the tool in an industrial environment.

In order to evaluate the applicability of the tool, the work presented in this Thesis had a clear criteria: **"it has to be useful in Industry"** (see Section 1.6). The research work presented is thus not about finding the perfect solution to deal with system evolution, but to provide a tool that is actually useful for the architect.

By the time of publication of this Thesis, the A3 Architecture Overview approach has already been incorporated into the Philips Healthcare MRI development process. From this fact we can conclude that the tool is applicable in an industrial context. The applicability of the tool is not limited to Philips Healthcare or MRI; other companies such as ASML, Océ, Daimler and FEI are testing the approach to incorporate it into their development process, due to the benefits the tool provides. We expect other companies to join in the near future.

In this chapter, we present experiences with the A3 Architecture Overview in an industrial context. Firstly, study cases in which the author created A3 Architecture Overviews to be used in real projects are presented. Lessons learned from those experiences, and the comparison of A3 Architecture Overviews and text documents is provided. Finally, to evaluate the A3 Architecture Overview creation effort, other study cases in which practitioners used an A3 Architecture Overview cookbook to create their own A3 Architecture Overviews for their projects are presented.

10.1 A3 Architecture Overview Study Cases

As in previous study cases (see Chapter 6), following the *Industry-as-Laboratory* approach (see Section 1.6) the author actively participated in different projects at Philips Healthcare MRI. In some of those projects, A3 Architecture Overviews were used to share and communicate architecture knowledge. Those projects are presented in Table 6.1.

Table 10.1: A3 Architecture Overview Study Cases

Philips MRI Projects
Control Communication Evolution (continuation)
New Style SDS (System Design Specification)
DDAS Network Architecture

In this section those projects will be described and the lessons learned from the application of A3 Architecture Overviews presented. Feedback from users of A3 Architecture Overviews collected from questionnaires will be used to evaluate the usage of the tool. In these projects, reverse architecting (see Chapter 7) was the approach used to create the A3 Architecture Overviews.

10.1.1 Project: Control Communication Evolution (continuation)

This project is the continuation of the project presented in Section 6.1.3. As the approaches used to capture and communicate architecture knowledge did not produce the desired results (see Section 6.1.3), those approaches were replaced by A3 Architecture Overviews.

Goal

Support the evolution of the control communication architecture by providing the experts with means to easily discuss alternative designs and requirements, enabling them to foresee potential impacts of those alternatives. Provide insight regarding the impact that a redesign in the communication architecture would have.

Outcome

Following the creation process as described in Chapter 9, system concerns were collected and a top level view of the control communication (from a system point of view) was developed (see Figure 9.2). To deal with time control, the MRI architecture design is based on "control words". Control words are bytes that provide control information to the different physical elements of the MRI. Therefore, from the top level view, main functions that deal with control words were transformed into A3 Architecture Overviews.

Three A3 Architecture Overviews were created for this project; *Generation of Control Words* (see Figure 10.1), *Distribution of Control Words* (see Figure 10.2) and *Store and Dispatch of Control Words*. The first one focuses on the creation of control words (byte stream) from the required waveforms in the MRI exam, the second one focuses on the distribution of those control words to the specific physical elements, and the third one on how those control words are consumed by those physical elements. From those A3 Architecture Overviews it is worth mentioning:

- *Scan Control* was identified as the system aspect that deals with control words. This therefore became the title of the A3 Architecture Overviews. The running title is the specific focus of each A3 Architecture Overview.
- Of special relevance for this system aspect is real-time control. To differentiate those functions or physical elements that needed real-time control or not, color coding was used (real-time control elements in orange, and non-real-time control elements in green).

- It can be observed in the A3 Summary of Figures 10.1, 10.2, that as the A3 Architecture Overviews focused on the same system aspect, the same top-level view was used.
- Although independent, the A3 Architecture Overviews are related as the outcome of the A3 Architecture Overview *generation of control words* (a byte stream) is the input to the A3 Architecture Overview *distribution of control words*.
- The *generation of control words* is implemented in software, therefore the physical view displays the software processes that are performed, while the *distribution of control words* focus on how the byte stream is distributed to the specific elements, therefore the physical view displays a network view of the system (both views are a decomposition of the top-level view).
- In *generation of control words*, in the quantification view, they key parameter presented is *number of control words* (per millisecond). As there was not a clear way to calculate the number of control words generated for a specific waveform, a figure of merit (formula) was created. In *distribution of control words*, the key parameters were *bandwidth* and *latency*. Again, a figure of merit was provided to calculate them.
- In the A3 Summary, additional sections were added. In *generation of control words* for example, how to deal with real-time control and other kind of control was described.
- As not each A3 Architecture Overview deal with all concerns, those system concerns that were not addressed in a particular A3 Architecture Overview were hatched in the system concern model (see Figures 10.1, 10.2)

The A3 Architecture Overviews were sent to the experts that deal with control communication to verify the information within the A3 Architecture Overviews and to provide them with the information about the expected design changes.

10.1.2 Project: New Style SDS

A **S**ystem **D**esign **S**pecification (SDS) specifies how system requirements are met (described in a System Requirement Specification (SRS) document). This specification relates the requirements of the system to the requirements of actual underlying "design elements". It was perceived by the SDS owner that despite the relevance of the SDS, it was seldom used, leading to development problems (this fact was confirmed by our survey, see Section 3.3.2).

Goal

A "New style SDS" initiative was launched to develop a new SDS, based on A3 Architecture Overviews, to capture and to display key system knowledge in a more effective way. The goal of the new SDS is to support MR development by providing a system overview that will improve communication across a broad set of stakeholders.

Outcome

Due to the relevance of the SDS within product development, and the drawbacks of the current SDS (see Section 4.2.2), it was decided that the A3 Architecture Overview style would be introduced as the new SDS style. A proof of concept SDS had to be developed to assess benefits and concerns of the new style.

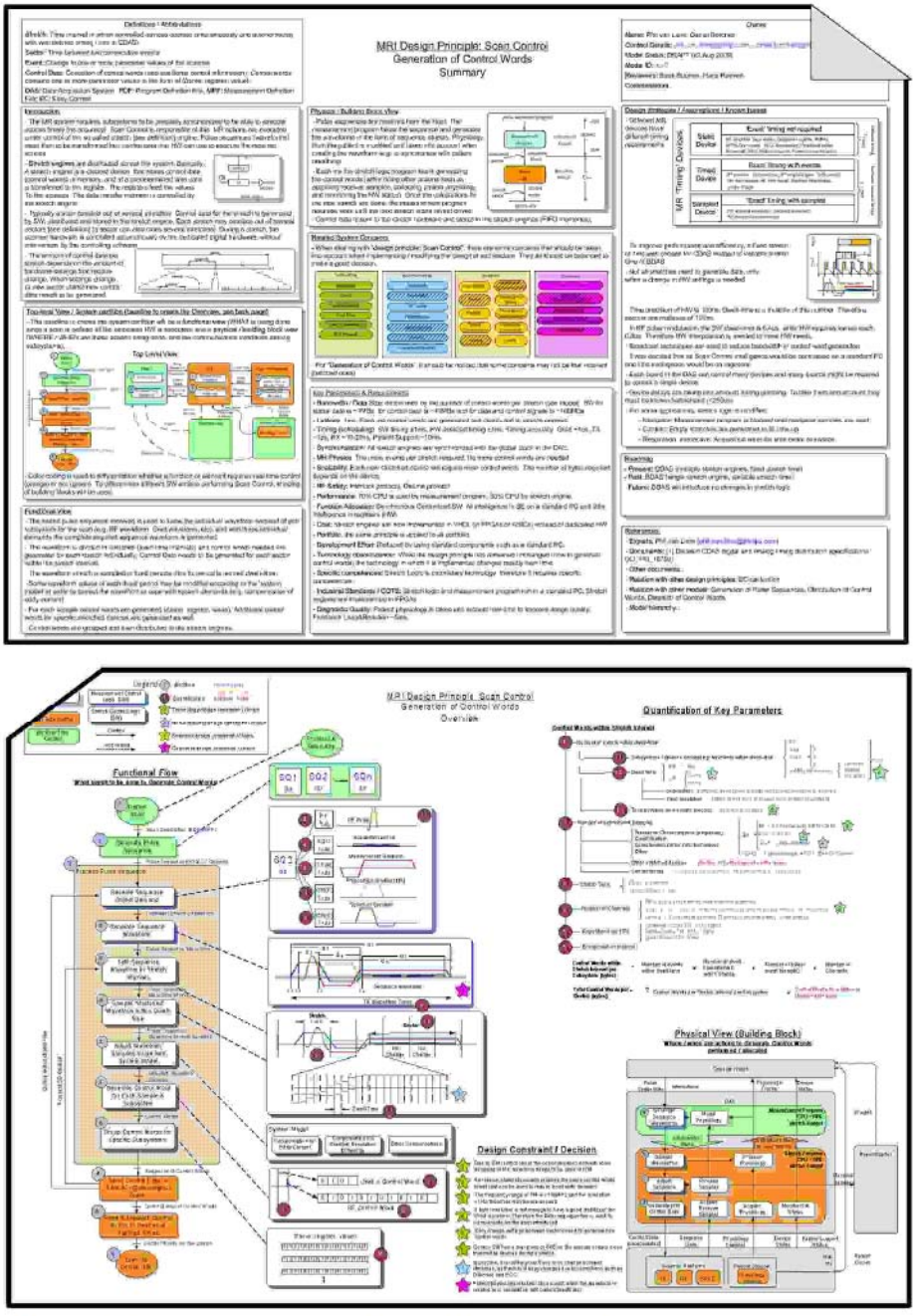


Figure 10.1: Scan Control - Generation of Control Words (non-readable due to confidentiality reasons)

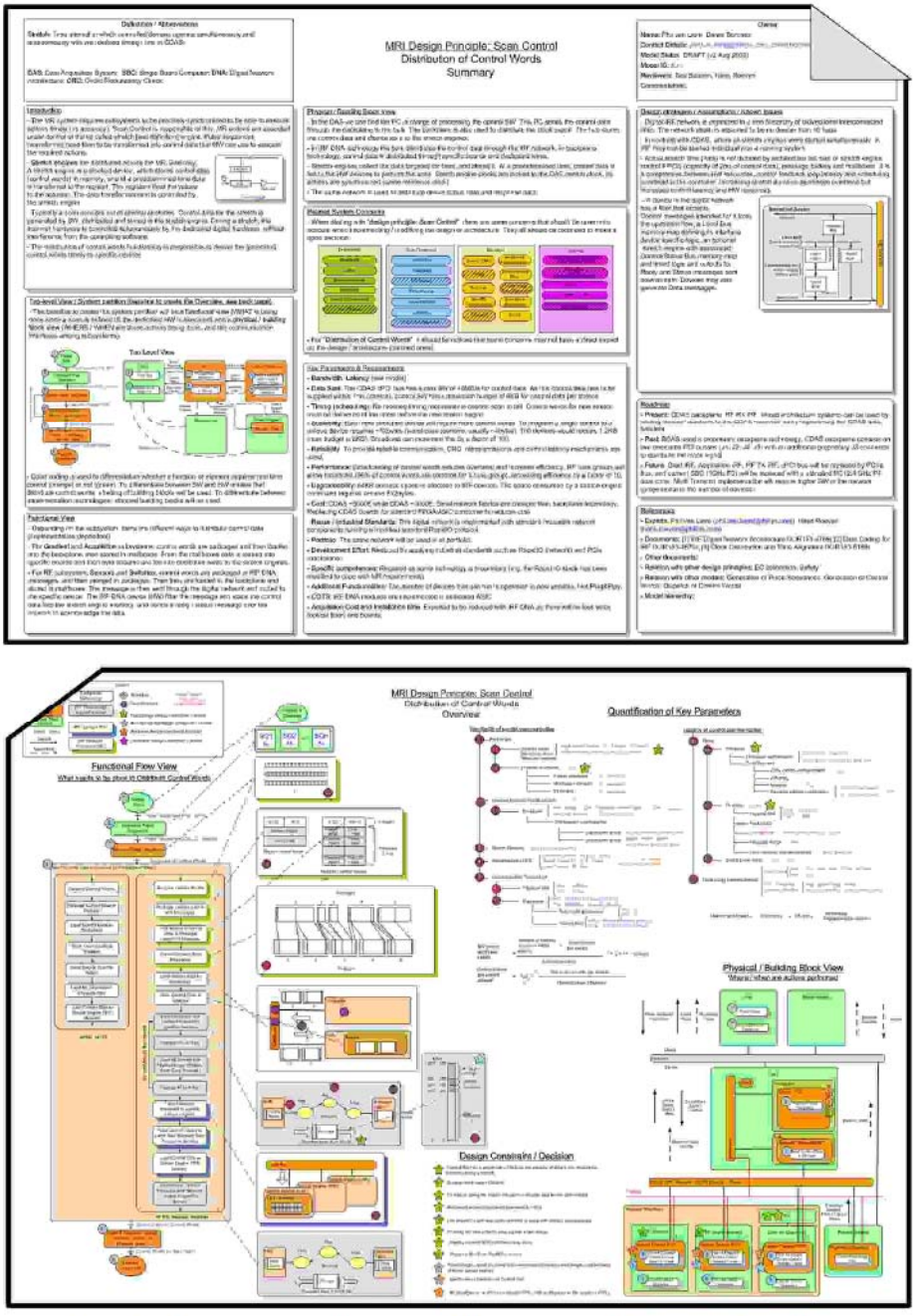


Figure 10.2: Scan Control - Distribution of Control Words (non-readable due to confidentiality reasons)

After an initial discussion with main MRI system architects, 16 system aspects[1] were identified as essential for the MRI system. It was estimated that each of those system aspects may need between one and four A3s once divided into topics. Then, an initial amount of 30-50 A3 architecture overviews were estimated to describe every relevant system aspect of the MRI system. This may look as a lot of work (previous SDS document was 200 pages long, created mainly by a few experienced individuals), however it should be noted that A3 Architecture Overviews describing a system aspect are created by different individuals or teams. Therefore the creation of a new SDS should be faster (as it can be done in parallel by different teams) and easier to update (as only A3 Architecture Overviews that change need to be updated[2]).

Two topics were selected for the SDS proof of concept; scan control and MRI calibrations. These system aspects were chosen as they were complex aspects that crossed system and organizational boundaries, resulting in interesting study cases.

- Scan control is in charge of controlling the different subsystems so they are executed in a timely and synchronized fashion. MRI systems need to have precise timing and be synchronized to the nanosecond, requiring complex solutions. The three A3 Architecture Overviews from the previous project were reused.

- MRI calibrations are in charge of correcting imperfections and tune different parameters for optimal image quality. Two A3 architecture overviews were created for two of the main MRI calibrations.

The SDS proof of concept was presented to the SDS owner and key stakeholders. The new style for the SDS was evaluated by the stakeholders through a survey (see Appendix A). After the proof of concept SDS was accepted and validated, a workshop with system architects and experts was organized to provide the means to complete the SDS. A short training in the creation of A3 Architecture Overviews was given, as well as some coaching by the author. To support the creation of those A3 Architecture Overviews after the workshop, an **A3 Architecture Overview Cookbook** was provided (which is an A3 Architecture Overview itself, as shown in Figure D.1). The A3 Architecture Overview Cookbook provides practitioners with an easy to follow guide to support the creation of A3 Architecture Overviews (as described in Chapter 9), and provides an A3 Architecture Overview example. The SDS owner is in charge of collecting the individual A3 Architecture Overview and keeping the hierarchy and relations among them.

10.1.3 Project: DDAS Network Architecture

DDAS is the evolution of CDAS (see Figure 2.5). CDAS is based on proprietary and obsolete backplane technology whereas DDAS introduces a network-based architecture (which will relieve CDAS limitations such scalability and will introduce new features). CDAS technology is reaching end-of-live limiting developments in other domains (e.g. limited number of channels) and its architecture requires a large number of cables. The introduction of DDAS will bring many design changes to the system at different levels.

[1] Stretch, Centralized SW, Digital Network Architecture, Calibrations, Scan Protocols, Security, Cooling, Power Distribution, Acoustic Noise, Power Management, Interactions, Safety, Interoperability, Scan Control, Multi-nuclei, Coil Control.
[2] How to detect which A3 Architecture Overviews need to change remains an issue.

Goal

To communicate the DDAS design to the variety of teams which will be involved in the developments the new design. Present the changes that DDAS will introduce relative to the CDAS design in order to visualize the impact of change.

Approach

The first challenge of this project was to explain the benefits of DDAS over CDAS and the need for changing the design to get management involved. In addition, it was essential for this project that the teams involved in the (future) DDAS development understand the DDAS design and the impact it causes to the system.

The design of the DDAS was mainly the work of a single system architect. For that reason, regular meeting were scheduled with him to collect, abstract and present the related architecture knowledge. In the initial discussion, as shown in Figure 10.3, it was agreed that the new design was meant to provide *scalability of system qualities* such as bandwidth. To realize the DDAS design, it was required to cope with two main aspects; *synchronization of network devices*, and *discovery and autoconfiguration of network devices*. For that reason, as shown in Figure 10.3, it was decided that four A3 Architecture Overviews should be created for this project.

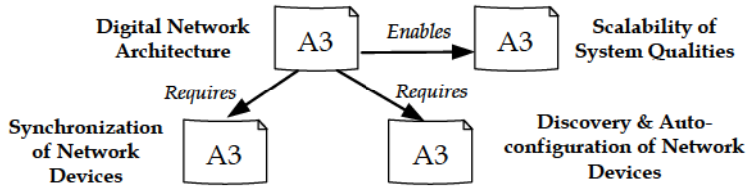


Figure 10.3: DDAS A3 Architecture Overviews

The A3 Architecture Overview *Digital Network Architecture* (see Figure 10.4) described the changes that DDAS brings to the system, as well the rationale for it, and the benefits this new design would bring, from a management perspective. From the A3 Architecture Overviews of this project is it worth mentioning:

- The A3 Architecture Overview *Digital Network Architecture* provides qualitative rather than quantitative estimations of the key parameters *scalability of MRI configurations* and*cost savings* (see Figure 10.4), while the A3 Architecture Overview *scalability of system qualities* provides tables with quantitative information.

- Each A3 Architecture Overview in this project does not share a common top level view, as each of them focuses on different system aspects, which are *Scalability*, *Synchronization*, and *Discovery and Autoconfiguration* respectively.

After the A3 Architecture Overviews were completed, they were sent to the different stakeholders to request feedback.

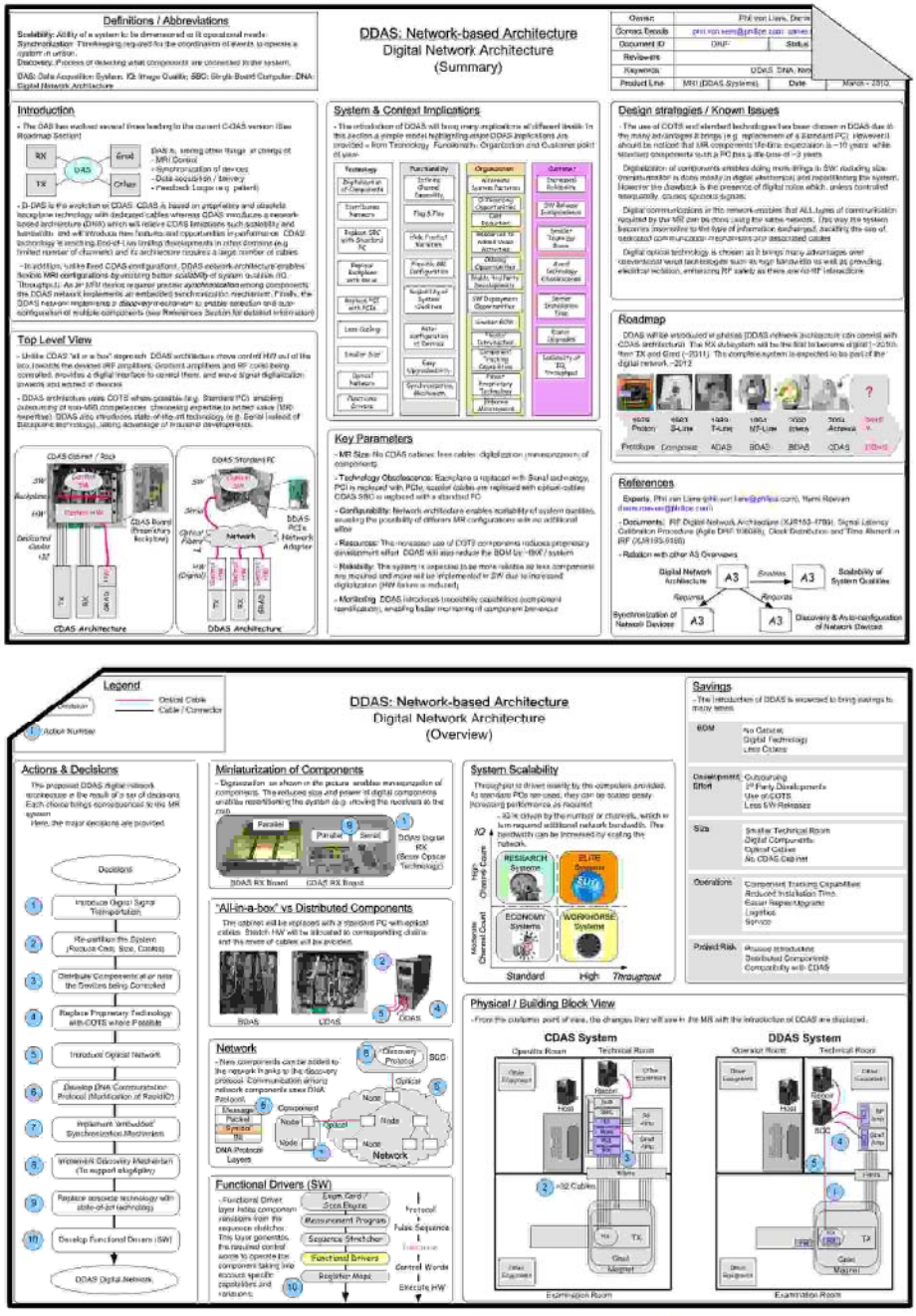


Figure 10.4: DDAS Network Architecture (non-readable due to confidentiality reasons)

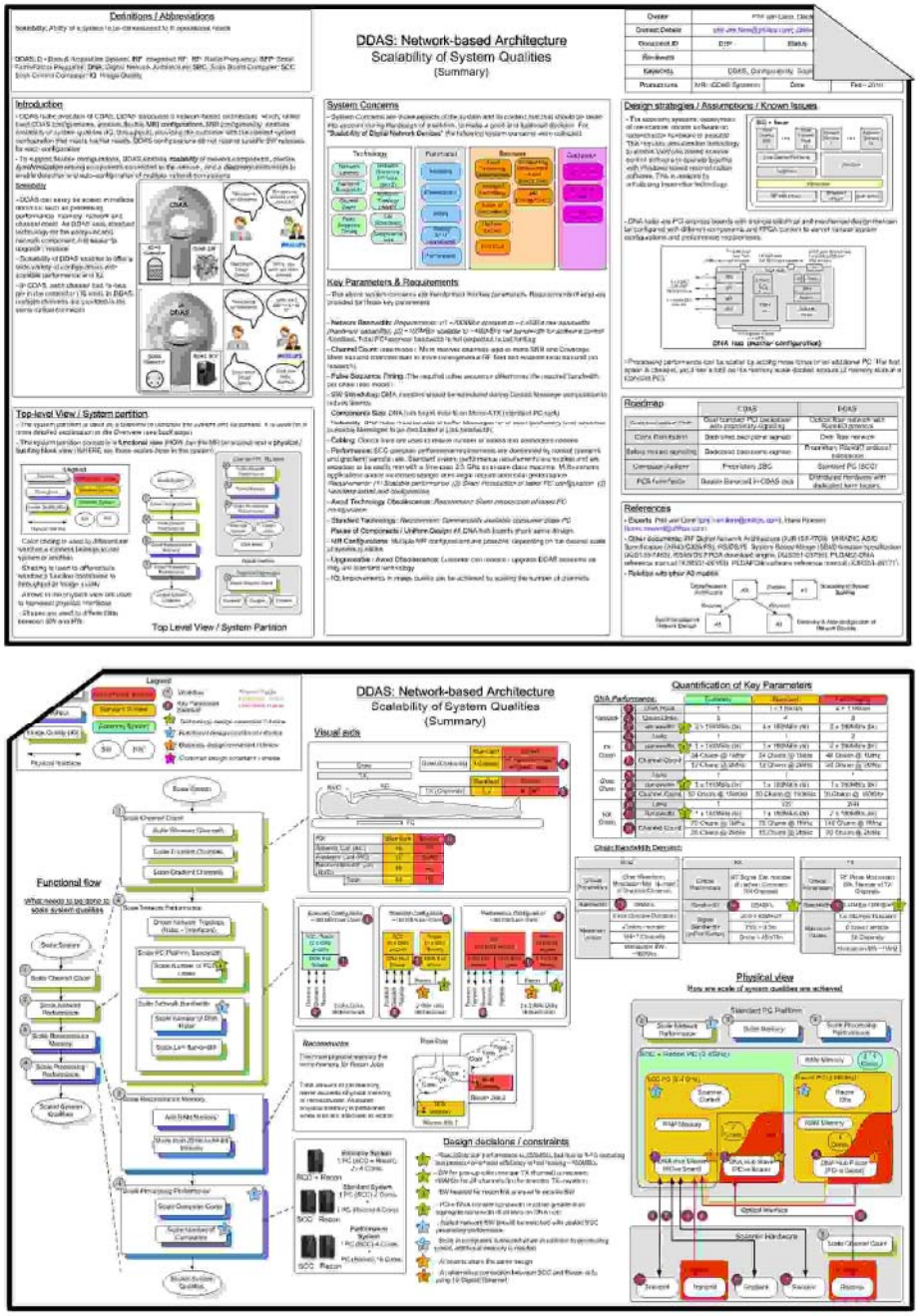


Figure 10.5: DDAS Network Architecture (non-readable due to confidentiality reasons)

10.1.4 Lessons Learned

It was stated by users that one of the main values of the A3 Architecture Overview is that it enables to get more insight into the system. This in itself has great value during the architecting process as *"one insight is worth a thousand analyses"* [Rechtin and Maier, 2000].

At design meetings, A3 Architecture Overviews were distributed beforehand to the members. The first reaction of those not used to an A3 was to question about the format chosen, i.e. hard to fit in the screen, problems with the printers, etc. However in the meetings we observed that all participants had read and studied the A3 provided, despite the remarks, while they did not read the equivalent text document that was also provided. This situation happened several times, leading to the conclusion that an A3 provides an amount of information employees are willing to read to prepare for a meeting or discussion.

People attended meetings with plenty of annotations in the A3 (see Figure 10.6), triggering discussions and improving the contents. The A3 Model supported discussions while the A3 Summary was barely used at meetings.

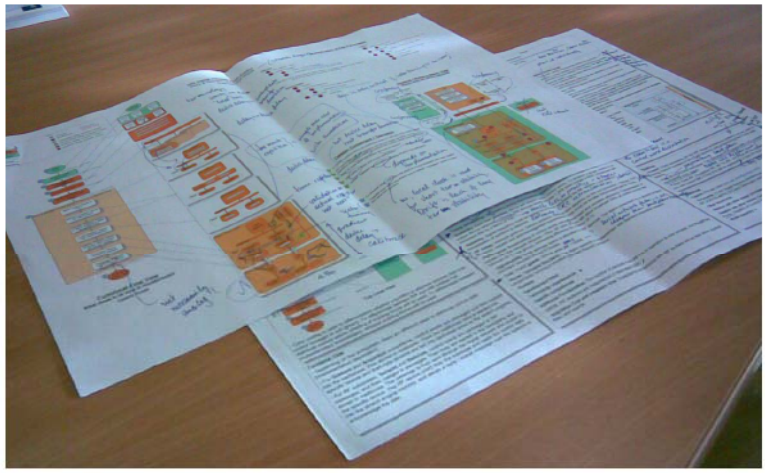


Figure 10.6: Capturing Architecture Insight with an A3 Architecture Overview

People from different disciplines and departments were able to use them without much explanation. Unlike in other projects in which documents or SysML models were used (see Section 6.1.3), discussions using A3 Architecture Overviews started right away. That is to say, we had less waste of time through more focused and productive meetings.

From the management point of view, it was stated that this new style improved maintainability and upgradeability of the design knowledge. New A3 Architecture Overviews could be added without having to touch the ones already created, and a new SDS could reuse existing A3 Architecture Overviews.

After the evaluation of this work, management decided to adopt this approach due to the benefits it provides. The approach has been adopted not just for the SDS, but also for capturing knowledge of system aspects that cross system and organizational boundaries. Currently senior designers are encouraged to create their own A3 Architecture Overviews.

Many users, after reading an A3 Architecture Overview, claimed that although it may look like a "limited" amount information, the density of the information contained is probably more than a regular document. That however did not prevent users from reading it. This showed that by providing information in this fashion, more information can be delivered to the reader without information overload.

10.2 Comparison of A3 Architecture Overviews with Traditional Documents

To compare A3 Architecture Overviews with traditional documents, practitioners were asked to rate properties of existing documents they use to acquire architecture knowledge (see Table A.10, Appendix A), as well as to rate the same properties of A3 Architecture Overviews they have used (see Table A.15, Appendix A). A comparison of the feedback given by practitioners is presented. From the analysis of the feedback, we can observe that, for practitioners:

- A3 Architecture Overviews are perceived as more readable than traditional documents, as show in Figure 10.7(a). Unlike traditional documents, no practitioner thinks an A3 Architecture Overview to have poor readability.
- A3 Architecture Overviews are perceived as easier to understand than traditional documents, as show in Figure 10.7(b).
- A3 Architecture Overviews are perceived as much more usable than traditional documents, as show in Figure 10.7(c).
- A3 Architecture Overviews have a more adequate amount of information than traditional documents, as show in Figure 10.7(d).

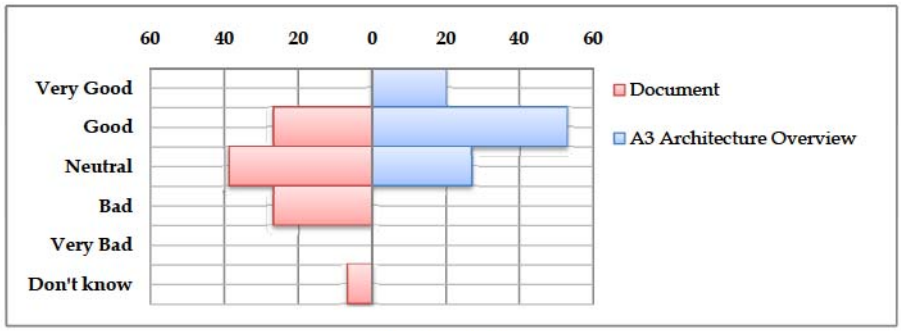
Regarding preferences, most practitioners (80%) stated that they would rather read an A3 Architecture Overview than a document. In addition, most practitioners (73%) stated that to learn a new topic an A3 Architecture Overview is better than a document (see Table A.13, Appendix A). From the feedback obtained through the survey, we observed that the more experience practitioners have, the more they prefer A3 Architecture Overviews over traditional documents.

10.3 Evaluation of A3 Architecture Overview Elements

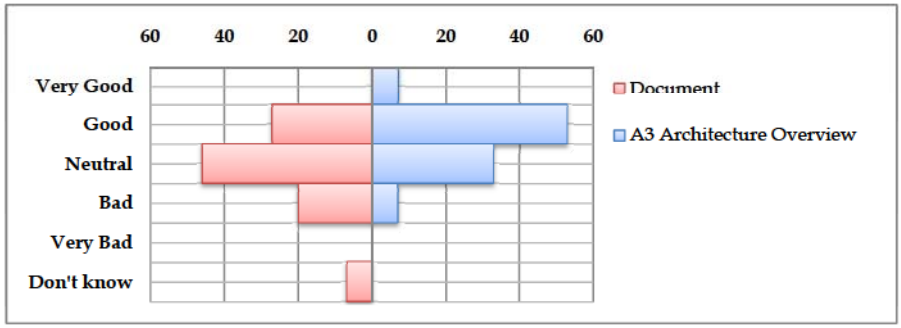
A3 Architecture Overview users were asked to rate how useful different elements used in the A3 Architecture Overview are. Based on their feedback (see Table A.12), we observed that more than half of the users found all elements of the A3 Architecture Overview useful or very useful.

As shown in Figure 10.8, annotations is the element of the A3 Architecture Overview that users found the least useful. This makes sense, as the annotations are meant just to highlight deviations from the ideal situation (e.g. due to business, technology or customer reasons), rather than providing a different view of the system.

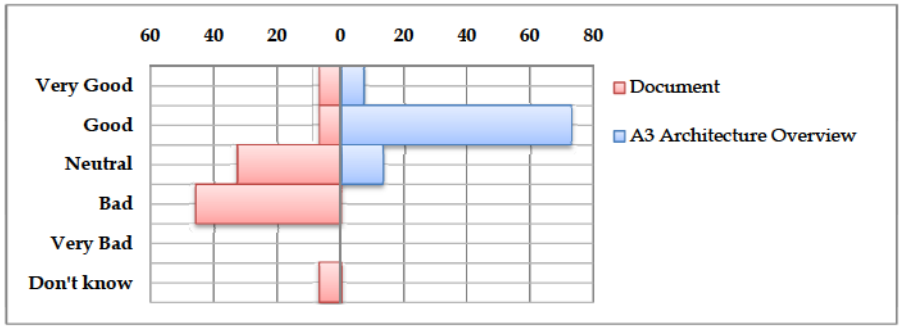
Some users also found the quantification view of little use. From the analysis of the survey it was found that those users were mostly engineers. The reason for this may be, as stated by [Rechtin and Maier, 2000], that engineers mostly deal with measurables using analytic tools derived from mathematics or other hard sciences. Architects on the other hand, deal largely with unmeasurables using approximations or estimations based on practical lessons learned.



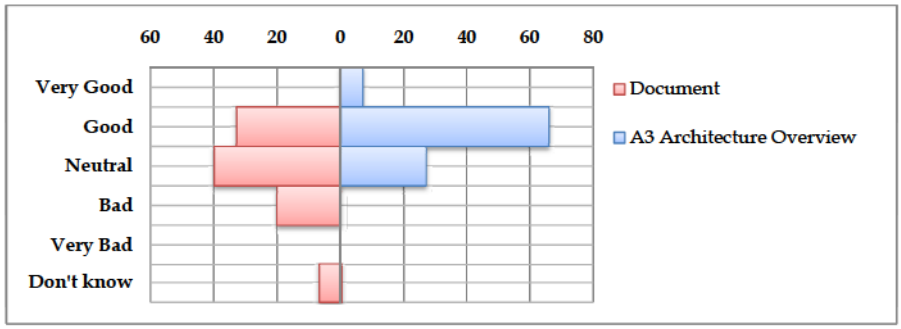
(a) Readability



(b) Understandability



(c) Usability



(d) Adequate amount of information

Figure 10.7: Comparison of traditional documents and A3 Architecture Overviews

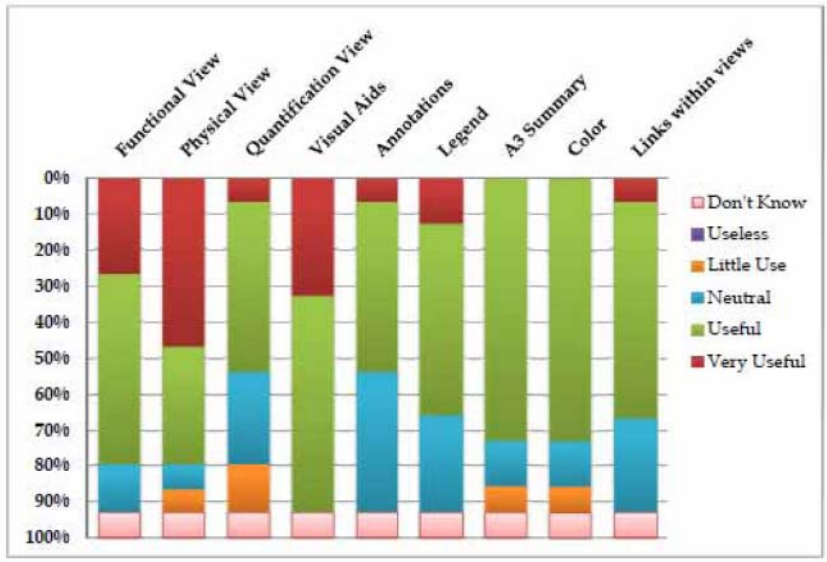


Figure 10.8: Evaluation of A3 Architecture Overview Elements

10.4 Other A3 Architecture Overview Study Cases

In this section we present other study cases in which A3 Architecture Overview were created by practitioners who showed interest in applying the tool in their projects. In these projects the author did not participate in the creation of the A3 Architecture Overview. Table 10.2 shows those projects in which project members expressed their interest in applying the A3 Architecture Overview approach.

Table 10.2: Other A3 Architecture Overviews Study Cases

Project	Author / Company	Title
Reconstruction Overview	Liere, P. (Philips MRI)	System Architect
Cooling Subsystem Evolution	Douglas, A. U. (Philips Research)	Philips Researcher
SAR (Specific Absorption Rate)	Laar, Pierre. (ESI)	Fellow Researcher
Litter Collector Robot	Kauw-A-Tjoe, R. G. (Twente University)	Master Student

To support the creation of A3 Architecture Overviews to people not familiar with the creation process, an A3 Architecture Overview Cookbook (see Appendix D) was provided as well as a few examples, to guide practitioners in the development process (as described in Chapter 9). Little additional support was provided by the author, so practitioners could create A3 Architecture Overviews freely.

In Figures 10.9, 10.11, 10.12, different A3 Architecture Overviews created by the practitioners from Table 10.2 are presented. We can observe that those A3 Architecture Overviews present different styles: the use of black&white, different fonts, a variety of A3 Summary layouts, the use of 3D images in the physical view, etc. Despite the differences, they can all be easily recognized as A3 Architecture Overviews.

To evaluate whether the A3 Architecture Overview could be created by people with no industrial experience, a master student was requested to create an A3 Architecture Overview for his final project. In Figure 10.12 the A3 Architecture Overview created by the master student is provided. There is no apparent need of industrial expertise to be able to create A3 Architecture Overviews.

10.4.1 Evaluation of A3 Architecture Overview Creation

To evaluate the effort required to create an A3 Architecture Overview and its value for the practitioners who created them, we conducted a survey to collect feedback from these practitioners (the survey details are provided in Appendix A.3).

Most of the surveyed reported that creating an A3 Architecture Overview is not difficult. They all agree that it is easier than creating an equivalent text document, and the value of the A3 Architecture Overview over a document is higher. For some practitioners, some of the steps in the creation were difficult, such as creating a functional view. However, all surveyed people agreed that the creation effort was not high for the benefits that having the A3 Architecture Overview brings.

All surveyed found that people that used their A3 Architecture Overviews showed interested in them, probably more than if they had provided a text document. Most of surveyed people (75%) thought that the A3 Architecture Overview helped to communicate better with the stakeholders. According to practitioners, none of the stakeholders who used their A3 Architecture Overviews had trouble understanding them. They all agreed that the information provided in the A3 Architecture Overview was enough for the stakeholders.

During the creation process none of the surveyed missed or needed the support of a software tool. They all agreed that having a predefined structure in the A3 was of help during the process, and they all saw the value of using an A3 layout instead of several A4s.

Most of them (75%) reported that they think that the approach is not difficult to transfer to other practitioners, and all of them stated that they will keep using the approach in future projects. They will also recommend the approach to other employees, as they reported that the approach is not hard to incorporate in their daily activities.

Strong points of the tool for authors — Practitioners stated the visual and graphical nature of the approach and the limited amount of information as strong points of the approach. The guidance provided by the structured approach in the creation process helped to provide an overview and to create uniform overviews across different subsystems. The approach forces people to be brief and having an A3 Architecture Overview triggers *"oh yeah, we must not forget..."* remarks. People stated that other areas besides architecture could benefit from the approach.

Weak points of the tool for authors — Practitioners stated that the weak point of the approach is that there is no guarantee that the creator of an A3 Architecture Overview will create a good one, as it is dependent on the person who creates it (which may lead to a

poor A3 Architecture Overview). Other concerns were related to some limitations of the paper format, such as cross referencing, and how to share A3 Architecture Overviews to the "electronic' community. People expressed their interest in enhancing the A3 Architecture Overviews capabilities by adding meta information such as hyperlinks and compatibility with simulation tools[3].

10.5 Discussion of the Results

To evaluate the applicability of the A3 Architecture Overview, this Thesis had a criteria: *"it has to be useful in industry"* (see Section 1.6). The tool has proven to be useful in different projects within Philips Healthcare MRI, and it has been incorporated into the daily development activities. In addition, the fact that other companies such as ASML, Océ, and FEI are evaluating it shows that the A3 Architecture Overview appears useful in industry.

Several study cases in which the A3 Architecture Overview has been used have been presented. We have learned that one of the main values of the A3 Architecture Overview is that it provides system insight, and that it provides an overview with sufficient information that people are willing to read for e.g. a meeting.

In comparison with traditional documents the A3 Architecture Overview is perceived as an improvement; it is more readable, easier to understand, more usable, and with more adequate information. Practitioners prefer an A3 Architecture Overview over a traditional document. Although we do not have enough representatives that have used the A3 Architecture Overview to derive hard conclusions, the observed trends and feedback make us confident that the A3 Architecture Overview can be used as a powerful tool at companies.

The evaluation of the individual elements of the A3 Architecture Overview shows that all elements are perceived as relevant, especially the visual aids. Annotations on the other hand seem to be the least useful for the users. Some views have more value for some disciplines than others, such as the quantification view. While architect were satisfied with the quantification provided, engineers were not.

Regarding the creation effort, the A3 Architecture Overview is perceived as easier to create than a document and to have more value. The creation effort is not perceived as high, and it can be easily incorporated into daily activities. Practitioners did not miss the support of a software tool, on the contrary, just the template provided by the A3 was enough to create A3 Architecture Overview.

[3] As discussed in Section 12.2, this is a topic of future research.

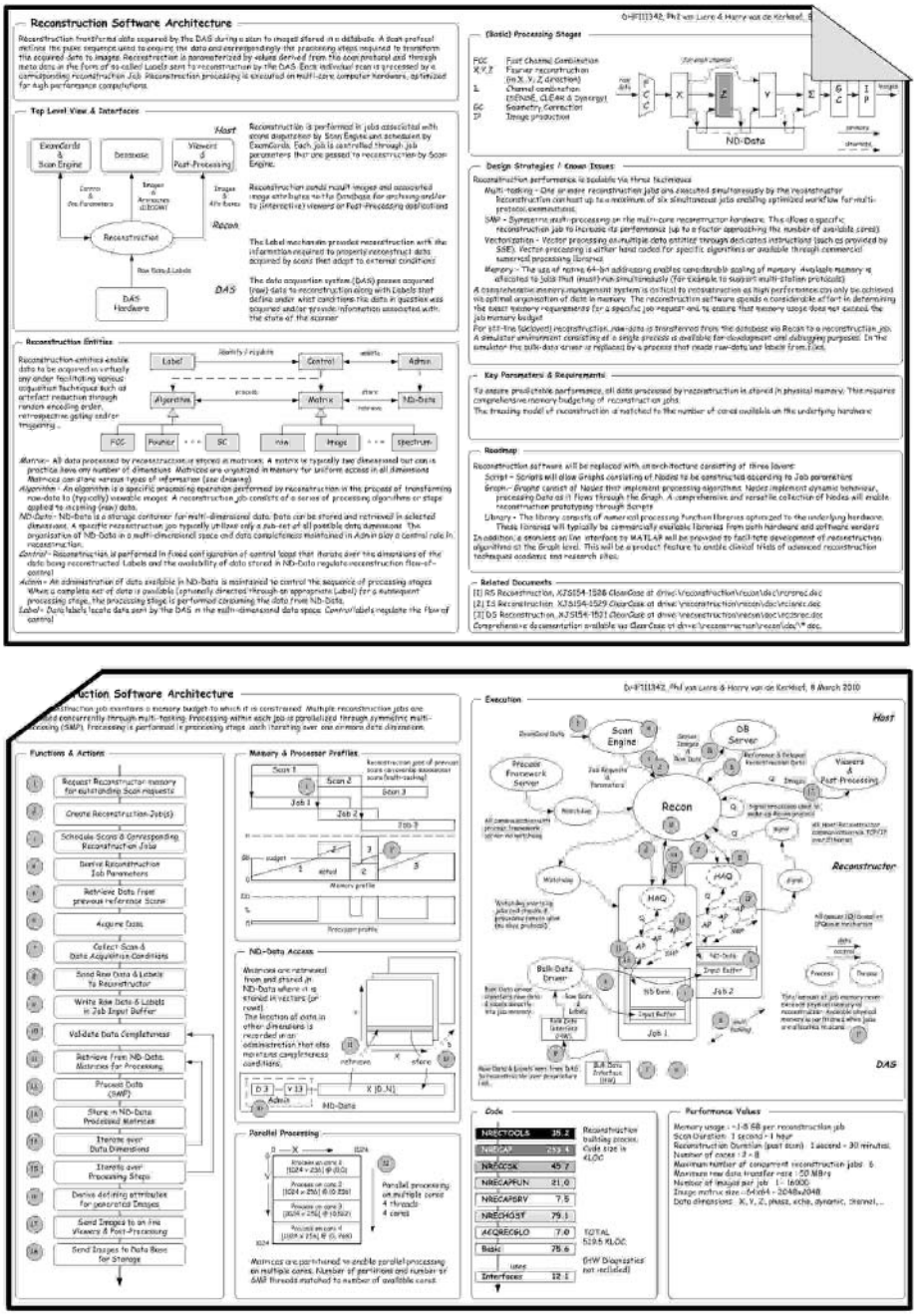


Figure 10.9: MRI Reconstruction SW Architecture (Courtesy of Phil van Liere) (non-readable due to confidentiality reasons)

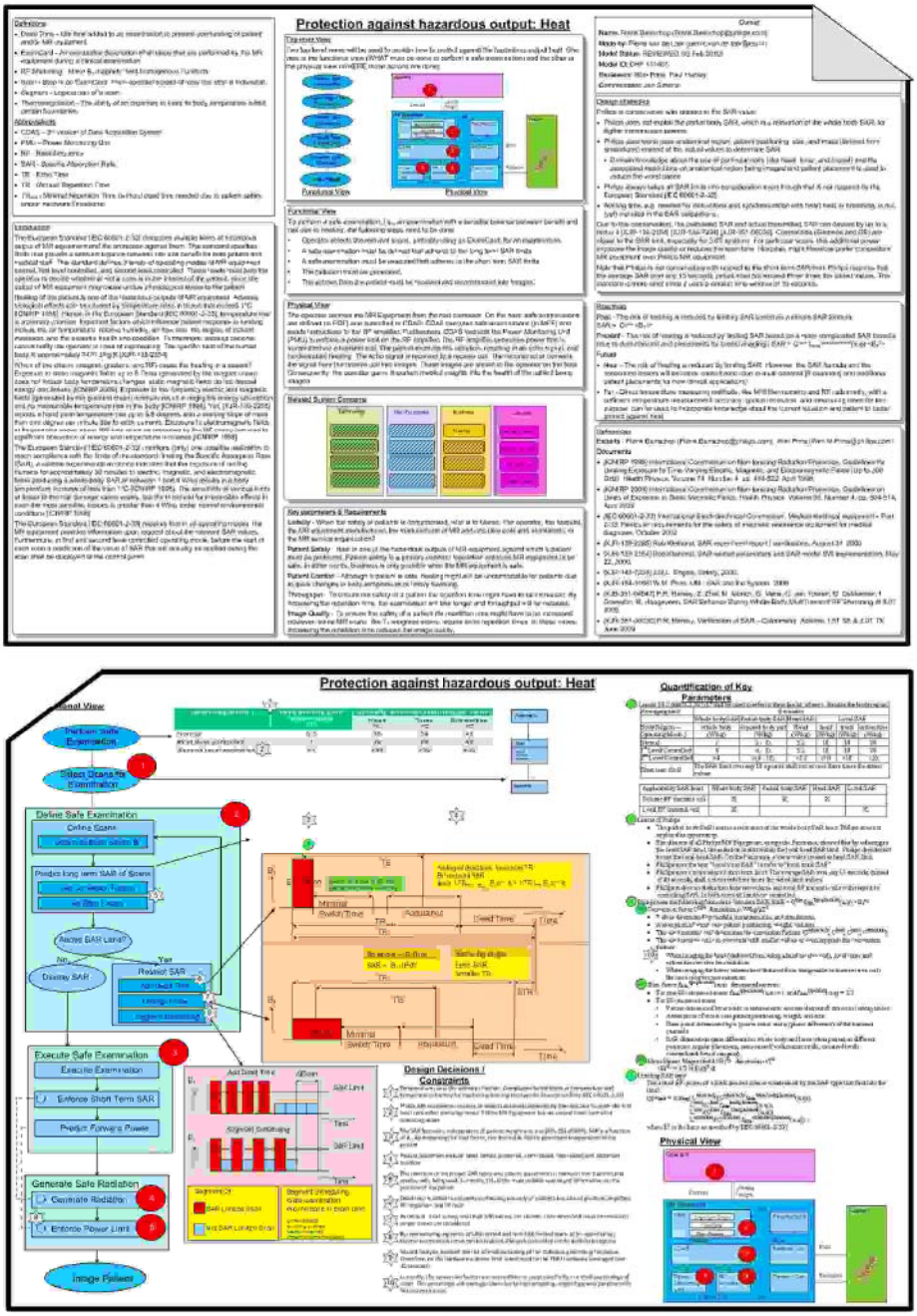


Figure 10.10: Protection Against Hazardous Output (Courtesy of Pierre van de Laar) (non-readable due to confidentiality reasons)

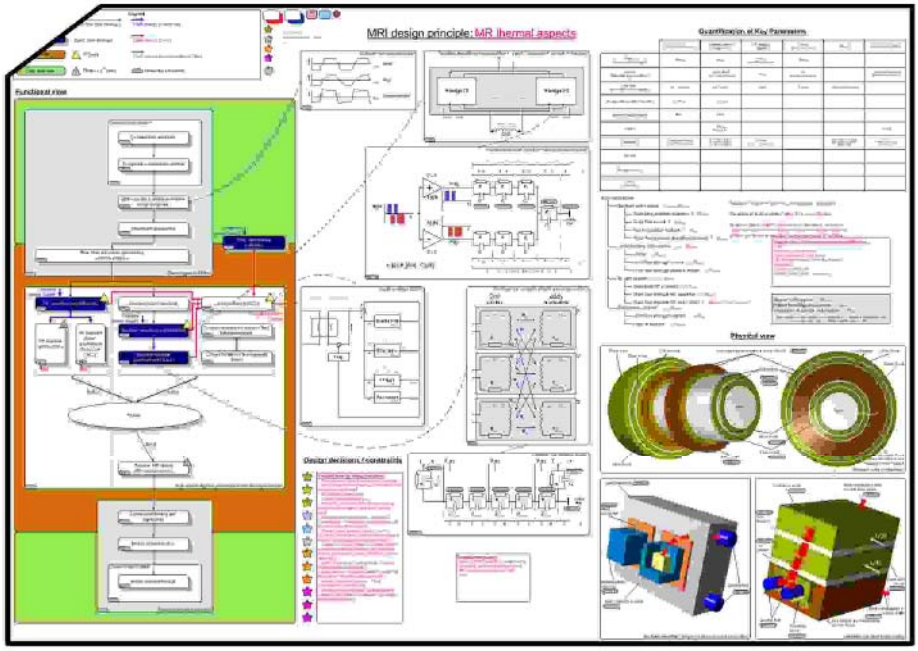


Figure 10.11: MRI Thermal Aspects (Courtesy of Alexander Douglas) (non-readable due to confidentiality reasons)

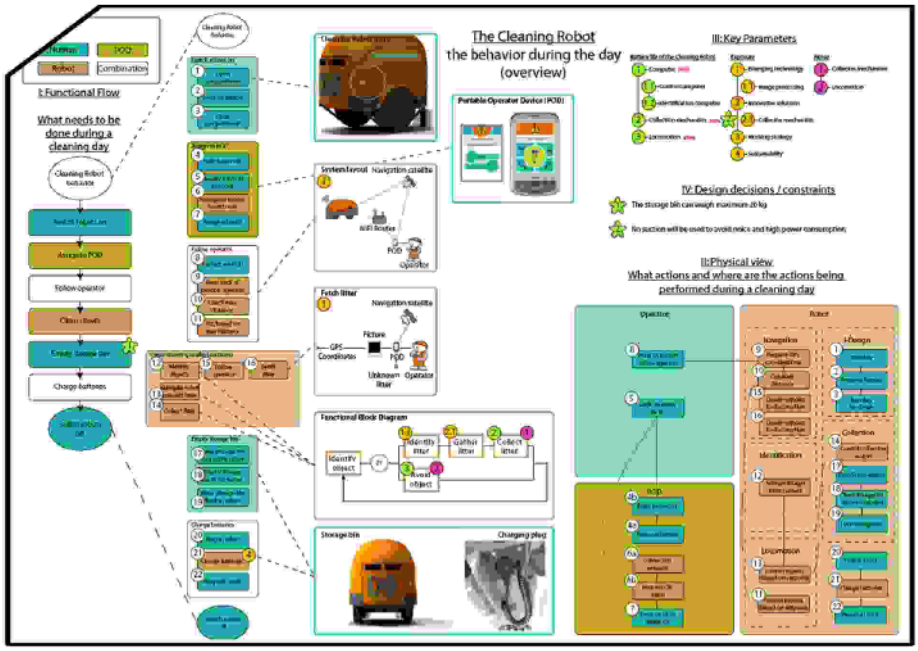


Figure 10.12: Cleaning Robot (Courtesy of Rogier Kauw-A-Tjoe)

Chapter 11

A3 Architecture Overviews as a Tool for Effective Communication in Product Evolution

In this chapter, the A3 Architecture Overview is evaluated as a tool for effective communication in product evolution. An evaluation based on a detailed analysis of the requirements identified in Part I of this Thesis is presented. Finally, a discussion about the work presented in this Thesis is provided.

The A3 Architecture Overview was designed as a tool to support effective communication in product evolution. To achieve that purpose, it has to meet a set of requirements. Those requirements were presented in Table 6.2. In this chapter, we evaluate whether the A3 Architecture Overview meets those requirements. For the evaluation, the detailed analysis of the requirements presented in Appendix B is used. The evaluation is based on direct experiences and observations of the use of A3 Architecture Overviews in an industrial environment, as well as feedback from users and creators collected through surveys (see Figure 11.1).

In Table 11.1 we present a summary of the requirements evaluation. In the evaluation, we have used the following criteria:

- **Good:** Based on the analysis performed (see Appendix B), we have clear indications that the A3 Architecture Overview meet the requirement.
- **Average:** Based on the analysis performed (see Appendix B), we have indications that the A3 Architecture Overview meet the requirement, however it may not improve existing means or there are some open issues.
- **Poor:** Based on the analysis performed (see Appendix B), we have clear indications that the A3 Architecture Overviewdoes not meet the requirement.
- **?:** There are not enough observations, experiences or feedback to support the evaluation of the requirement, and the evaluation is based on some assumptions.

11.1 Requirements Evaluation

Practical Industrial Requirements of Communication Tools

From Table 11.1, we can conclude that the A3 Architecture Overview meets most of the practical industrial requirements that communication tools should have; it requires a small overhead, it does not depend on custom-made software tools, it works even with incomplete input, it is easy to use, and it is appealing to most users.

We can observe, however, that the A3 Architecture Overview does not guarantee trusted output. Although the A3 Architecture Overview encourages providing confidence levels (e.g. for the quantification), provides references to experts and documents, etc, the credibility of the information contained in the A3 Architecture Overview depends on the person who creates it. In this sense, there is no difference with traditional documents.

Table 11.1: Evaluation of A3 Architecture Overviews as a Tool for Effective Communication in Product Evolution

Requirements	**Evaluation**			
I Meet Practical Industrial Needs of Communication Tools	**Good**	**Average**	**Poor**	**?**
Require small overhead	X			
Do not depend on custom-made software tools	X			
Provide trusted output		X		
Work even with incomplete input	X			
Easy to use	X			
Appealing	X			
II Meet Desired Properties of Communication Tools	**Good**	**Average**	**Poor**	**?**
Provide limited amount of information	X			
Use visual representations	X			
Use an appropriate size to display complex information	X			
Keep the notation simple		X		
Limit the amount of visual attributes and ensures differences among them	X			
Keep a consistent way of communication	X			
Provide a shared view	X			
Enable a flexible way to share knowledge	X			
Improve existing written mechanisms	X			
III Be Tailored to the Architecting Process	**Good**	**Average**	**Poor**	**?**
Support the creativity of architects		X		X
Used by a wide variety of stakeholders	X			
Do not take too much time from architects	X			X
Encourage the dissemination of knowledge	X			
IV Support the Needs of Architects	**Good**	**Average**	**Poor**	**?**
Deliver the right information to the stakeholders while keeping the irrelevant part of information low	X			
Ensure that the information is conveyed and interpreted correctly		X		
Record changes in the architecture knowledge repository			X	X
Retrieve architecture knowledge stored in the heads of people	X			
Enable reusing knowledge from previous experiences and products in current developments	X			
Help keeping a structured overview of what has been communicated with a stakeholder	X			
V Mitigate Evolution Barriers	**Good**	**Average**	**Poor**	**?**
Support management of system complexity			X	X
Prevent the lack of system overview	X			
Deal with ineffective knowledge sharing	X			
Help finding the required system information	X			
Support communication across disciplines and departments	X			
VI Deal with Observed Evolution Challenges	**Good**	**Average**	**Poor**	**?**
Support moving from incremental development to top-down architecting		X		X
Reduce learning curve	X			X
Support to estimate the impact of change	X			X
Deal with the mono-disciplinary focus of developers	X			
Support repartitioning the system		X		X

Desired Properties of Communication Tools

From Table 11.1, we can conclude that the A3 Architecture Overview meets most of the requirements of desired properties of communication tools; it provides a limited amount of information, it uses visual representations, it uses an appropriate size to display complex information, it limits the amount of visual attributes and ensures differences among them, it keeps a consistent way of communication, it provides a shared view, it enables a flexible way to share knowledge, and it improves existing written mechanisms.

Regarding the notation, although the A3 Architecture Overview aims to keep it simple by using simple visual attributes (e.g. blocks, arrows) and natural language, it does not satisfy all stakeholders. Most disciplines found the notation not complicated, however half of the architects stated it may be complicated. As architects are expected to be the main users of the tool, this raises a concern. However, it was stated by most stakeholders, including the architects, that A3 Architecture Overview are not difficult to understand (despite the notation). This means that, at least for some architects, the A3 Architecture Overview notation is not easier than other communication tools (e.g. documents).

Tailored to the Architecting Process

From Table 11.1, we can conclude that the A3 Architecture Overview is tailored to the architecting process to a large extent, as it can be used by a wide variety of stakeholders, it does not take too much time from architects, and it encourages the dissemination of knowledge.

We do not have enough evidence that the A3 Architecture Overview supports the creativity of architects, at least not in a direct way. The A3 Architecture Overview however does not constrain it, as the A3 Architecture Overview is flexible enough to adapt to preferred styles of capturing knowledge.

Although experiences in the creation of A3 Architecture Overviews by the author and other practitioners (see Table 10.2) indicate that it does not require much time (less than a traditional document) and that it can be easily incorporated into the daily activities, we do not have enough architect representatives to ensure it does not take an excessive amount of time. More experiences with architects are needed.

Support the Needs of Architects

From Table 11.1, we can conclude that the A3 Architecture Overview meets many of the architects needs, but fail to meet some of them. The A3 Architecture Overview delivers the right information to the stakeholders while keeping the irrelevant information low, it retrieves architecture knowledge stored in the heads of people -along with the reverse architecting process-, it enables reusing knowledge from previous experiences and products in current developments, and helps keeping a structured overview of what has been communicated with a stakeholder.

An A3 Architecture Overview encourages the use of visual representations to ensure that the information is conveyed and interpreted correctly, however as an A3 Architecture Overview contains natural language, the model notation may be ambiguous (unlike modeling languages such as SysML, see Section 5.2.1), this may lead to some interpretation problems. This means that in this requirement the A3 Architecture Overview has not improved existing mechanisms such as traditional documents.

Regarding recording changes in the architecture repository, there are some issues that makes it difficult to meet this requirement. As stated by some practitioners, cross referencing among A3 Architecture Overviews may lead to problems when changing the repository; a change in one A3 Architecture Overview may require to update other A3 Architecture Overviews (e.g. the reference section), leading to some problems. As this is a manual process, required changes may be overlooked if the responsible of the repository is not aware of them.

Mitigate Evolution Barriers

From Table 11.1, we can conclude that the A3 Architecture Overview meets most of the requirements to mitigate evolution barriers, such as to prevent the lack of system overview, to deal with ineffective knowledge sharing, to help finding the system information, and to support communication across disciplines and departments.

Regarding management of system complexity, A3 Architecture Overviews provide a manageable set of information. By partitioning the system knowledge in small doses, the A3 Architecture Overview tries to help managing system complexity by making it easier to digest the large amount of information to the user. System complexity, however, is complicated as it involves many domains (see Section 2.3.1). As the A3 Architecture Overview focuses on architectures, it does not deal with the other domains that make the system complex.

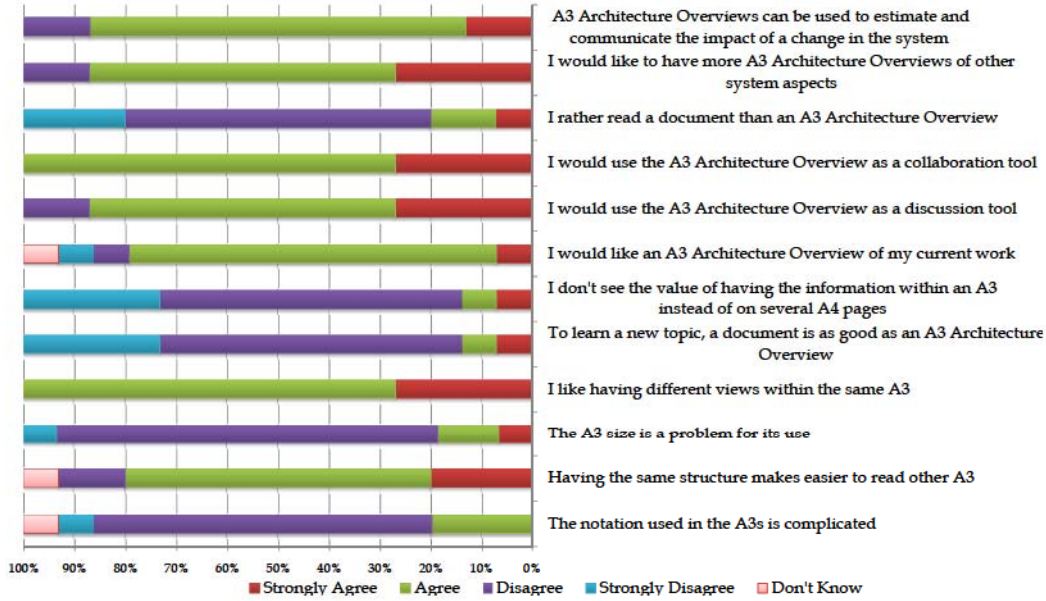
Deal with Observed Evolution Challenges

From Table 11.1, we can conclude that the A3 Architecture Overview meets some of the requirements to deal with observed evolution challenges, such as to reduce the learning curve, to support the estimation of the impact of change, and to deal with mono-disciplinary focus of developers.

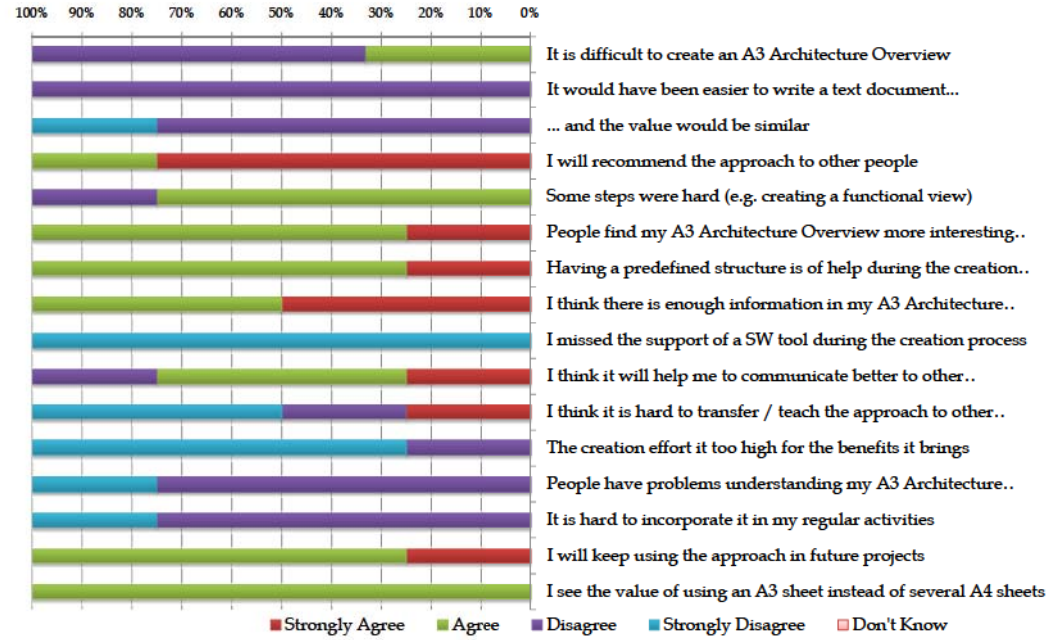
Although we cannot prove directly that an A3 Architecture Overview reduces the learning curve, the A3 Architecture Overview has been pointed by users as better way for learning about a new topic than other means such as documents. It was stated by users that they prefer to read an A3 Architecture Overview than a document, especially when learning about a new topic. In addition, a visual representation is usually more effective for learning than plain text, as it is retained better by the reader [Koning, 2008]. As an A3 Architecture Overview encourages the use of visual representations, it is probably better suited for learning than pure text documents. According to the users, the visual aids within the A3 Architecture Overview are the most valuable element.

Although we have not used the A3 Architecture Overview to estimate the impact of change in a real project, an A3 Architecture Overview provides different visual representations of the system (e.g. functionality, physical), and provides numbers to estimate the impact to the key parameters, which can be used to estimate the impact of change. Most A3 Architecture Overview users stated that the A3 Architecture Overview can be used to estimate the impact of change.

Finally, even when having an explicit architecture representation may support to move from incremental development to top-down architecting or to repartition the system, it is not clear that an A3 Architecture Overview deal with those challenges better than other means such as traditional documents. More experiences with the use of the tool are needed to evaluate this.



(a) A3 Architecture Overview User's Feedback (Source: Appendix A.2)



(b) A3 Architecture Overview Creator's Feedback (Source: Appendix A.3)

Figure 11.1: Feedback from A3 Architecture Overview users and creators

11.2 Discussion

This section will review the concept of system evolution and systems architecting, and the need for knowledge sharing and effective communication. Several observations will be made. We will also reflect briefly on the results obtained with the A3 Architecture Overview tool and the experiences of its application in an industrial environment.

11.2.1 System Evolution and Systems Architecting

In Chapter 3 we showed that current market demands are driving companies to adapt their development and design processes to cope with e.g. shorter development schedules, product optimization, cost reduction, etc. The strategy often chosen to cope with those demands it to evolve existing products, so infrastructure, knowledge and experience can be reused in the development of a new product. However, evolving a complex system is not a trivial task, as unexpected problems may arise due to dependencies, known and hidden, in the system and the organization. In this sense, an evolvable system, one that enables easy evolution, is vital for the success of companies.

We found that although in other research communities such as computer science software evolution is an active field of research, in the Systems Engineering field, system evolution is almost an unexplored field. In this Thesis we have reviewed the concept of system evolution and evolvability, and explored existing approaches to deal with system evolution. The first thing we have observed is that research usually addresses the system evolution problem by looking at the system alone and neglecting the influence of the design process; that is to say, either the system is designed to withstand changes or not, and that determines its evolution capability. It is therefore not surprising that most research work regarding system evolution focuses on the creation of modular designs (as a modular design is supposed to be more evolvable), or approaches to estimate the impact of a design change (as the less impact, the easier to evolve a system).

We believe that the ability of a system to be easily evolved, **evolvability**, is not just a property of the system's design, but also of its design process. To support this claim, when investigating development and evolution barriers, we found that major problems companies face when evolving products happen during the development and evolution process. Barriers such as managing system complexity, communication problems among disciplines and departments, lack of a shared system view, etc, are human-related aspects that affect the evolution process rather than problems in the system's design. For this reason, focus of this Thesis is not on redesigning a system to be more evolvable, but on providing the means to support the design process so systems can be evolved more efficiently and more effectively.

The fact that evolvability is seeing in literature as a property to withstand change, instead of a property aimed to cope with -inevitable- change, has lead to confusion regarding its differences with other system properties such as adaptability, changeability or extensibility. In this Thesis, an evolvable system is not just a system designed to adapt to change, but one in which the -multidisciplinary- team(s) in charge of the evolution process can easily identify what aspects of the design can be reused to develop a new product, and what new aspects need to be incorporated to meet the new demands. In an evolvable system, the team(s) would be aware of the impact that the required design changes would have on the system, so adequate strategies can be put in place to avoid undesired impacts.

It is worth mentioning, that the identified evolution barriers are usually already known development problems in any company (e.g. Philips management was aware of most of those barriers). They are however taken as a fact of life, which is hard to change and therefore little effort is spent on solving them. We have shown the relation between those barriers - especially ineffective knowledge sharing- and development problems and poor decisions. Once this relation was identified and quantified, more attention was given to the problem. By addressing those barriers system evolution can be enhanced.

If we are not looking at the design, how can we help the people in charge to evolve the system? As stated before, an evolvable system -or its context- provides the means for the team(s) to easily identify what system aspects can be reused to develop a new system. However, during the evolution process, technologies, implementations and even designs may have changed. In this context, what can be reused? The answer lies in the architecture. By utilizing an architecture as a framework to support multiple (successive) implementations, one ensures that it is not likely to change. Therefore, architectures should be captured so they can be used as a means to support the design of new generations of systems. The creation of architectures and their consolidation in architecture representations is part of the architecting process. The architecting process results in a series of major design decisions that shape the system as well as the evolution process. As systems architecting plays an essential role in the evolution of systems, this Thesis focuses on architectures and systems architecting.

11.2.2 Sharing and Communicating Architecture Knowledge

As we discussed in Chapter 4, architecting remains an art which is typically performed by people who have gained experience and knowledge over the years; the **architects**. Architects are responsible for the evolution process, as they make key decisions that will affect the system and its development process. In this Thesis we have reviewed the duties of architects, and what needs they have in order to support their work. By understanding their needs, we are in a better position to develop means to support architects in their duties. In this Thesis we showed that main needs of architects are related to sharing and communicating architecture knowledge. Architects need means to support tasks such as to recover implicit architecture knowledge spread within the company and to ensure that the architecture knowledge is delivered and communicated effectively to the stakeholders. It is for this reason that in this Thesis we focus in developing means to share and communicate architecture knowledge.

Sharing Architecture Knowledge

As described in literature, we have also experienced that architects prefer simple approaches over complicated (even when more complete or apparently more useful) approaches, they do not like to rely on tools, and have a preference for easy to use solutions. We have observed that most research solutions aimed to support architects in their duties are developed from a technology perspective, such as providing "intelligent advice", automated architecting analysis, etc. Research and our own experience has shown us that architects rather prefer to stay in control of the architecting process. We have also observed that architects prefer to get the facts and make their own conclusions, rather than trusting the outcome of a tool.

Once we know that we need to support sharing and communicating architecture knowledge, one of the challenges is to identify what types of information belong to the architecture knowledge. Most approaches to capture architecture knowledge focus on the architecture structure, in the sense of components and connectors. However for complex systems,

other types of information are needed. From literature, we observed that different communities have different needs of architecture knowledge, such as functionality, design decisions, problem domain, etc. An inspection of architecture documents, revealed that most of those types of information are usually present, yet not clearly differentiated. We have gathered those essential types of information that should be part of the architecture knowledge. It could be argued that we may have left out some essential type of information, however in our definition of architecture knowledge, we have left room for information *"that is important for the stakeholders"* [Fowler, 2003]. By gathering those types of information as part of the architecture knowledge, we aim to cover the range of essential architecture information that most stakeholders need.

When looking at existing approaches to share architecture knowledge, we observe that the effort is placed on capturing and delivering as much information as possible, but little effort is put on how this information is communicated to the stakeholders. Despite existing contributions, current practice is that architects, for good reasons, choose their own way to capture architecture knowledge. Traditional text documents such as design specifications still remains as the common choice to capture this knowledge. In this Thesis we have evaluated the effectiveness of traditional documents to share and communicate architecture knowledge, by evaluating Philips MRI System Design Specification (SDS). We found that the document not an effective mechanism to share architecture knowledge. These results were shared with other companies, which stated similar problems. However, the fact that traditional documents are not an effective means to share knowledge, does not mean that we have to move from traditional text documents to state-of-the-art technology-based solutions. We believe that there is still room for improving existing written mechanism, and that taking a smaller jump from existing means that users can digest is a better approach.

Effective Communication, a Basis for Knowledge Sharing

Even if an approach enables to share architecture knowledge, that is to say, all relevant architecture information is captured and delivered to the stakeholders, this does not mean the approach is automatically successful. If knowledge is not effectively communicated, that is to say, correctly understood by the stakeholders, it can render the approach ineffective. Effective communication is essential for knowledge sharing. Effective communication in a multidisciplinary environment is, however, a challenge. Not surprisingly communication among disciplines and departments was identified as one of the main barriers for system evolution. For effective communication is necessary to understand the -human- communication process in order to develop effective mechanisms to share knowledge.

Although ubiquitous, the human communication process is very complex as it involves many domains. For that, it is out of the scope of a single Thesis to cope with all those domains in detail. To model the communication process in the systems architecting context in a way that is easy to understand, we have used the Shannon-Weaver model. This simple model has enabled us to identify the main elements that take part in the communication process. By using this model, major "architecture noise" sources that affect the communication process have been identified. Although in this Thesis we cannot cope with all kind of noise sources, we aim however to cope with major noise sources produced by human and organizational factors. By doing this, we aim to develop an approach to consolidate architecture knowledge in a way that enables sharing of knowledge in a fashion that supports effective communication.

When looking at existing approaches to deal with communication in the Systems Engineering field, we find Model-Based Systems Engineering (MBSE). MBSE is a well-known

strategy to support the design of systems in a way that supports communication. There are many ways in which MBSE can be applied; from the development of simple informal models formed of simple free-format views, to the more formal and systematic by using modeling languages such as SysML. We have used different MBSE approaches in real projects at Philips Healthcare, to find out what works and what does not in the system architecting context. We found, as described in Chapter 6, that some MBSE approaches such as using functional and physical views were useful when communicating with stakeholders, while too many views, the use of documents, modeling languages, or automated approaches such as DSM were not successful. Lessons learned from the application of different approaches in real projects at Philips Healthcare MRI, as well as findings from literature were used to understand the requirements that an approach meant to support product evolution should meet.

11.2.3 A3 Architecture Overviews and Reverse Architecting

Architects need a way to capture and share existing architecture knowledge, in a way that ensures that it is effectively communicated to the stakeholders. For that, a process to make the architecture knowledge explicit and a tool to consolidate and share that knowledge in a way that supports effective communication are needed. In this Thesis we have proposed a **reverse architecting** process for recovering and making explicit the existing architecture knowledge, and the **A3 Architecture Overview** as the tool to capture that knowledge in a way that supports effective communication.

The creation of the A3 Architecture Overview is the result of applying practical experiences over theories. Practical aspects have been applied rather than formal or complicated approaches. With this tool, we do not aim to solve all the evolution problems but to provide a useful tool that the practitioner architect can use when evolving a system. By looking at the results of this work we believe we have succeeded in that.

For the physical design of the tool, we chose an A3 sheet size. This way we do not force a technology solution, we limit the amount of information provided, we provide a size to display knowledge that fits well within the average human field of view, etc. In the A3, we have separated model and text. While the A3 Model aims to provide visual representations for shared understanding (e.g. at meetings), the A3 Summary aims to provide the necessary text to support the model and to make the A3 Architecture Overview an independent document. We found that the A3 Model alone ended up embedded in text documents losing its utility. Although the finding of the adequate size to display complex information (A3) was found through experiments, we found later that Toyota had already reached the same conclusion to make their reports more efficient. Although they are used for a completely different purpose, the goal of using an A3 is the same; keeping only essential information.

The elements to display in the A3 are the different types of information that belongs to the architecture knowledge. An appropriate representation, encouraging visual and easy to understand representations has been chosen for each of those elements. The structure within the A3 is aimed to support readability; by providing a consistent structure across A3 Architecture Overview, the readers can find the information more easily. The power of the approach lies in:

- It forces brevity and synthesis of knowledge, providing limited amount of information but of high quality.
- It is visual in nature, structured, and appealing, supporting communication between different stakeholders.

- It is easy to use as no domain knowledge or experience is needed, encouraging sharing of knowledge.
- It does not depend on software tools or technologies.

To bring together the reverse architecting process and the A3 Architecture Overview, we provide a step-wise guide to guide people to consolidate architecture knowledge in an A3 Architecture Overview. This step-wise approach was consolidated in an A3 Architecture Overview Cookbook (see Appendix D), which is an A3 Architecture Overview itself, providing the guide as well as an A3 Architecture Overview example. This A3 Architecture Overview Cookbook was the primary thing provided to people to enable them to create A3 Architecture Overviews themselves. It is a powerful tool to transfer the approach to other practitioners.

11.2.4 Experiences and Results

The A3 Architecture Overview has been applied in real projects at Philips Healthcare MRI in order to evaluate its applicability in an industrial context. Both its use and its creation effort have been evaluated. Experiences, observations and surveys have been used for the evaluation.

By using A3 Architecture Overviews in projects we learned that they are perceived as a good way to get insight in the system, and they are useful as a collaboration and communication tool. People attended the meetings in which A3 Architecture Overviews were provided having read the A3 Architecture Overviews provided and with plenty of annotations in them. People from different disciplines were able to use them without much explanation.

If we compare the A3 Architecture Overviews with traditional documents, we have seen that they are perceived by users as being more readable, easier to understand, much more usable, and with a more adequate amount of information. This shows that the A3 Architecture Overview has some advantages over traditional documents when dealing with architecture knowledge.

From the experiences of practitioners in the creation of A3 Architecture Overviews, we have learned that it is not difficult to create (except possibly some parts) and that it does not require much time to create -less than a traditional document-. Practitioners stated that people who used their A3 Architecture Overviews found them interesting and were able to understand them without problems. They also stated that they will keep using the approach in future projects and recommend it to other people. This indicates that practitioners not only value in the use of the A3 Architecture Overview, but they also see the value in their creation.

Regarding its applicability in an industrial context, the A3 Architecture Overview has already been incorporated into the daily activities of Philips Healthcare MRI, while several other companies are testing it. We can conclude that it is applicable in an industrial context.

We cannot claim we have improved evolvability, as we do not have a clear way to prove that in the time available. The improvement of evolution process may take years to evaluate, and there is not an easy way to prove a direct relation between the application of the tool and the success or failure of the evolution. However, by looking at the evaluation of the tool, we see that the A3 Architecture Overview meets most of the requirements needed for a tool to support effective communication in product evolution. This shows the potential utility of the A3 Architecture Overview to support product evolution.

Chapter 12

Conclusions and Recommendations

In this final chapter the main conclusions obtained from the research carried out in this Thesis are provided. Finally, recommendations to continue the work started in this Thesis to trigger future research related to the A3 Architecture Overview are presented.

12.1 Conclusions

Companies need to shorten their development cycles while delivering new functionalities and optimized products. As creating a product from scratch is time consuming and costly, the strategy often chosen is to evolve an existing product. Yet evolving a product is also a challenging endeavor. The impact of change in the product's design can ripple through the system's design leading to development problems. An evolvable product, one that facilitates evolution, would clearly specify which aspects of the design should be passed down to and which are new to the previous generation, and would keep the impact of change under control. Most of those strategies are borrowed from the software field. There are only a few specific strategies to deal with evolution at system level, which are either limited, not applicable in practice, not adequate to complex systems, or do not fit in an industrial environment.

Complex systems such as MRI scanners introduce a new dimension to the evolution problem, as people from multiple disciplines need to collaborate to integrate the many aspects involved in its design and development. It is therefore not surprising that major evolution barriers are managing system complexity, difficulties in estimating the impact of change, lack of system overview, ineffective knowledge sharing, and communication problems. All those barriers are mainly related to the people who creates and modifies the system rather than the system itself. By supporting those people with the right means, we can facilitate the evolution process, thus enhancing evolvability.

To deal with evolution, architectures are needed. During the evolution of a system, technologies, implementations and the design may have changed, it may be only the architecture that remains from the original system. Architectures, once consolidated in an architecture representation, capture essential knowledge that is needed for the evolution of systems. There are many ways in which architectures can be captured, however what knowledge should be captured and how to represent it to support the evolution of complex systems is unclear. Based on our experiences with different approaches to represent architectures in an industrial environment, we propose an architecture overview to represent the architecture knowledge.

The creation of architectures and architecture representations is part of the architecting process, and that responsibility lies on the architect. One of the major needs of architects is to share architecture knowledge, and ensure that this knowledge is communicated effectively. Existing solutions to support architects usually do not meet those needs, are not tailored to the architecting process, or are not optimal for an industrial environment. Most solutions are developed from a technology perspective, while architecting is mainly a human activity. Architecting solutions need to focus more on the human side of architecting.

Besides the need of sharing architecture knowledge to support evolution, it is necessary to ensure that the knowledge is communicated effectively, it is to say, understood by the stakeholders. Many approaches to share knowledge do not take into account that human and organizational factors affect communication, and consequently, although effective in theory, when applied in a real situation where people is involved, they are not accepted. Knowledge sharing requires effective communication, therefore, any mechanism designed to share knowledge should take into account communication aspects to avoid the "noise" that may affect the communication process.

In this Thesis we have designed a tool, the A3 Architecture Overview, meant to share architecture knowledge in a fashion that enables effective communication in an industrial context. To that end, human and organizational factors are taken into account. The A3 Architecture Overview is not meant to be complete, formal or executable, it is meant to be an artifact that the practitioner architect can use during his daily activities.

An A3 Architecture Overview is not meant as the perfect solution to cope with evolution, but an improvement over traditional means to share and communicate architecture knowledge. Experiences of its use in real projects in a company such as Philips Healthcare MRI has proven that it meets the requirements to become an effective communication tool during product evolution. Proof of that is that the A3 Architecture Overview has become part of the daily development activities within Philips Healthcare MRI, and that other companies such as ASML, FEI, Daimler, EADS and Boeing are testing it.

The main value of an A3 Architecture Overview lies in its simplicity. It is an effective way to capture architecture knowledge yet it does not require long hours of specialized training to create or use it. An A3 Architecture Overview delivers the essential knowledge that is needed in product evolution in a format that encourages sharing of knowledge and in a fashion that can be understood by a wide variety of stakeholders. It may not be formal, complete, executable, or supported by state-of-the-art technology, but unlike other approaches, it is not put aside but it is used by current practitioners.

12.2 Recommendations

The A3 Architecture Overview is still in an early stage. There is much to do, the current A3 Architecture Overview has only scratched the surface of what can be done with it. Many users and creators have stated *"can we use an A3 Architecture Overview in..."*, *"what if we change the A3 Architecture Overview to.."*, and similar remarks. This shows that there is room for more research regarding the A3 Architecture Overview.

During the realization of the projects in which the A3 Architecture Overviews where used as well as during discussions at different companies or symposiums (INCOSE), the following areas of improvement have been identified:

- Incorporation of meta-data and hyper link capabilities: When A3 Architecture Overviews are displayed in a screen, it would be desirable that the content within the A3 Architecture Overview enable to navigate to other content (e.g. hyper text, clickable objects from the views, etc). For that, an A3 Architecture Overview would need to embed meta-data and enable hyper link capabilities. This would also enable searching capabilities within A3 Architecture Overviews contents. To this end, it is necessary to investigate how to incorporate this meta-data and how to enable the hyper link capabilities. It should be noticed that this should not require additional effort during the A3

Architecture Overview creation process to prevent making it more complicated (e.g. by adding the meta-data and hyper links to finished A3 Architecture Overviews).

- Enhanced cross-referencing: A set of A3 Architecture Overviews for a repository of architecture knowledge. When a change in one A3 Architecture Overview is required or a new A3 Architecture Overview is created, this may require some updating to existing A3 Architecture Overviews from the repository (e.g. updating the reference section). Currently this change has to be done manually. It would be desirable to investigate how a repository (or other means) could automatically update existing A3 Architecture Overviews affected by a change or incorporation in the repository.

- Application of the A3 Architecture Overview in other domains besides architecting: The A3 Architecture Overview is designed to display architecture knowledge. The A3 Architecture Overview, however, could probably be applied to other domains as well, such as design, engineering, etc, as the need of overview is not exclusive to architecting. To tailor the tool to other domains, it would be necessary to research the essential knowledge that the A3 Architecture Overview should provide for those domains, the adequate way to visualize that knowledge, and an appropriate structure of the A3 to display the visualization.

- Link between the A3 Architecture Overview and other tools: It would be beneficial to verify and validate the A3 Architecture Overview contents by e.g. running simulations. From that, it would be necessary to develop a way to transform the A3 Architecture Overview notation (which is not formal and up to the A3 Architecture Overview creator) into another notation that can be fed into an executable tool.

- Creation support: Currently white boards and papers are used to draw the sketches that will be part of the A3 Architecture Overview. There are many devices in the market such as table PCs, Microsoft surface computers, touch screens, or smart white boards, that could enable the creation of views, without the need of transforming the sketch into a digital view. It would be interesting to investigate whether the use of those devices in the creation of A3 Architecture Overviews would enhance (or diminish) the efficiency of the creation process.

- Use in agile developments: Agile methods such as Scrum or Extreme Programming break tasks into small increments with minimal planning. Iterations are short time frames (timeboxes) that typically last from one to four weeks. Each iteration involves a team working through a full development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders. In this context the A3 Architecture Overview could be applied. If would be interesting to investigate how the agile methods could benefit from the incorporation of a tool like the A3 Architecture Overview.

Bibliography

Allen, P. N. and C. M. John: 2005; *An approach to the measurement of software evolution: Research Articles*; **J. Softw. Maint. Evol.**; **vol. 17** (1): pp. 65–91.

Altenberg, L.: 1994; *Advances in Genetic Programming*; *Tech. rep.*; MIT.

Altshuller, G.: 1998; 40 Principles: TRIZ Keys to Technical Innovation; Technical Innovation Center, INC.

Ariyanto, D.: 2010; *Material F1 Usage Accuracy*.
`http://dodik1.blogspot.com/2010/05/mid-review-internship-program.html`

Avgeriou, P., P. kruchten, P. Lago, P. Grisham and P. Dewayne: 2007; *Archiectural knowledge and Rationale - Issues, Trends, Challenges*; in SHARK'07.

Avison, d., F. Lau, M. D. Myers and P. A. Nielsen: 1999; *Action Research*; **Communication of the ACM**; **vol. 1**: pp. 94–97.

Axelsson, J.: 2002; *Towards an Improved Understanding of Humans as the Components that Implement System Engineering*; in 12th International Symposium of the International Council of Systems Engineering (INCOSE'02); Las Vegas, USA.

Babar, M. A. and I. Gorton: 2007; *A Tool for Managing Software Architecture Knowledge*; in 2nd Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent (SHARK/ADI).

Baldwin, C. and K. Clark: 2000; Design rules: The power of modularity; MIT Press, Cambridge.

Baskerville, R.: 1999; *Investigating Information Systems with Action Research*; **Communication of the Association for Information Systems**; **vol. 2**: pp. 1–31.

Bass, L., P. Clements and R. Kazman: 2003; Software Architecture in Practice; Addison-Wesley.

Bensley, E., L. Fisher, M. Gates, J. Houchens, A. Kanevsky, K. Soohee, P. Krupp, A. Schafer and B. Thuraisingham: 1995; *Evolvable Real-Time C3 Systems*; **First IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'95)**; p. 153.

Berelson, B. and G. Steiner: 1964; Human behavior: An inventory of scientific findings; New York: Harcourt, Brace, and World.

Berko, R. M., A. Wolvin and D. Wolvin: 2003; Communicating; Houghton Mifflin Harcourt (HMH); 9th edition.

Berlo, D.: 1960; The process of communication: An introduction to theory and practice; New York: Holt, Rinehart and Winston.

Bertin, J.: 1983; Semiology of Graphics; University of Wisconsin Press.

de Boer, R. C. and R. Farenhorst: 2008; *In search of 'architectural knowledge'*; in SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge; pp. 71–78.

Bonnema, G. M.: 2008; Funkey architecting : an integrated approach to system architecting using functions, key drivers and system budgets; Ph.D. thesis; University of Twente.

Bonnema, G. M. and P. D. Borches: 2008; *Design with Overview - How to Survive in Complex Organizations*; in Proceedings of the 18th Annual Symposium of the International Council of Systems Engineering (INCOSE'08); Utrecht, The Netherlands.

Bonnema, G. M. and F. J. A. M. Houten: 2006; *Use of Models in Conceptual Design*; **Engineering Design**; **vol. 17** (6): pp. 549–562.

Borchers, J., O. Dussen and A. Klingert: 1996; *Layout Rules for Graphical Web Documents*; **Computer&Graphics**; **vol. 20**.

Borches, P. D. and G. M. Bonnema: 2008a; *'Living' Architecture Overviews - Supporting the Design of Evolutionary Complex Systems*; in CIRP Design Seminar; Twente, The Netherlands.

Borches, P. D. and G. M. Bonnema: 2008b; *On the Origin of Evolvable Systems: Evolvability or Extinction*; **Proceedings of the TMCE**; **vol. 2**: pp. 1351–1353.

Borches, P. D. and G. M. Bonnema: 2009; *Coping with System Evolution- Experiences in Reverse Architecting as a Means to Ease the Evolution of Complex Systems*; in Proceedings of the 19th Annual Symposium of the International Council of Systems Engineering (INCOSE'09).

Borches, P. D. and G. M. Bonnema: 2010a; *A3 Architecture Overviews, Focusing Architectural Knowledge to Support Evolution of Complex Systems*; in Proceedings of the 20th Annual Symposium of the International Council of Systems Engineering (INCOSE'10).

Borches, P. D. and G. M. Bonnema: 2010b; *System Evolution Barriers and How to Overcome Them!*; in Proceedings of the 8th Annual Conference on Systems Engineering Research; pp. 455–464.

Boucher, M. and D. Houlihan: 2008; *System Design: New Product Development for Mechatronics.*; *Tech. rep.*; Aberdeen Group, Boston, MA.

Browning, T. R.: 2001; *Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions*; **IEEE Transactions on Engineering Management**; **vol. 48(3)**: pp. 292–306.

Bunge, M.: 1977; Treatise On Basic Philosophy: The Furniture of the World; Springer, Reidel, Dordrecht.

Capilla, R., F. Naval and J. C. Dueñas: 2007; *Modeling and Documenting the Evolution of Architectural Design Decisions*; in 2nd Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent (SHARK/ADI).

Chan, S. Y.: 2001; *The use of graphs as decision aids in relation to information overload and managerial decision quality*; **Journal of Information Science**; **vol. 27(6)**: pp. 417–425.

Chijiiwa, H.: 1987; Color Harmony; Rockport Publishers.

Chikofsky, E. J. and J. H. Cross II: 1990; *Reverse Engineering and Design Recovery: A Taxonomy*; **IEEE Software**.

Christian III, J.: 2004; *A Quantitative Approach to Assessing System Evolvability.*
`http://ntrs.nasa.gov/search.jsp?Ne=35&N=123+126+4294888785`

Christian III, J. and J. Olds: 2005; *A Quantitative Methodology for Identifying Evolvable Space Systems.*
`http://hdl.handle.net/1853/8375`

Clarkson, P. J., S. Caroline and E. Claudia: 2004; *Predicting Change Propagation in Complex Design*; **Journal of Mechanical Design**; **vol. 126** (5): pp. 788–797.
`http://link.aip.org/link/?JMD/126/788/1http://dx.doi.org/10.1115/1.1765117`

Clements, P., R. Kazman, M. Klein, D. Devesh, S. Reddy and P. Verma: 2007; *The Duties, Skills, and Knowledge of Software Architects*; in WICSA, 6th Working IEEE/IFIP Conference on Software Architecture.

Cook, S., H. Ji and R. Harrison: 2000; *Software evolution and evolvability*; *Tech. rep.*; UK.

Crawley, E., O. Weck, S. Eppinger, C. Magee, J. Moses, W. Seering, D. Schindall and D. Whitney: 2004; *The Influence of Architecture in Engineering Systems.*
`http://strategic.mit.edu/PDF_archive/4%20Other20Major%20Pubs/4_5_ESD2004_architecture-b.pdf`

Cummings, J.: 2003; *Knowledge Sharing: A Review of the Literature*; *Tech. rep.*; The World Bank Operations Evaluation Department.

Daft, R. L. and R. H. Lengel: 1984; *Information richness: a new approach to managerial behavior and organizational design*; **Research in Organizational Behavior**; **vol. 6**: pp. 191–233.

Darwin, C.: 1859; On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life; Hurst, London.

Dasgupta, S.: 1991; Design theory and computer science: processes and methodology of computer systems design; Cambridge University Press.

Dawkins, R.: 1981; *In Defence of Selfish Genes*; **Philosophy**; **vol. 56**: p. 556573.

Domb, E. and K. Radeka: 2009; *LAMDA and TRIZ: Knowledge Sharing Across the Enterprise*; **TRIZ Journal**.

Eeles, P.: 2006; *The Process of Software Architecting*; *Tech. rep.*; IBM.
`http://www-128.ibm.com/developerworks/rational/library/apr06/eeles/index.html`

Eppler, M. J.: 2004; *Knowledge Communication Problems between Experts and Managers*; in ICA Working Paper, University of Lugano.

Eppler, M. J. and J. Mengis: 2003; *A Framework for Information Overload Research in Organizations.*; in Insigts from Organization Science, Accounting, Marketing, MIS, and Related Disciplines (ICA Working Paper, University of Lugano).

Ernst, F. and P. S. Armin: 2005; *Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle*; **Syst. Eng.; vol. 42** (4): pp. 135–152.

Fan, W., L. Wallace, S. Rich and Z. Zhang: 2006; *Tapping the Power of Text Mining*; **Communication of the ACM; vol. 49(9)**: pp. 77–82.

Farenhorst, R., J. F. Hoorn, P. Lago and H. van Vliet: 2009; *What Architects Do and What They Need to Share Knowledge*; *Tech. rep.*; VU University, Amsterdam.

Farenhorst, R., P. Lago and H. van Fliet: 2007a; *Prerequisites for Successful Architectural Knowledge Sharing*; in Proceedings of the 2007 Australian Software Engineering Conference (ASWEC'07).

Farenhorst, R., P. Lago and H. van Vliet: 2007b; *Effective Tool Support for Architectural Knowledge Sharing*; in 1st European Conference on Software Architecture, Ed. F. Oquendo, Springer LNCS 4758; pp. 413–437.

Fill, C.: 1999; Marketing Communications, Context, Contents and Strategies; Prentice-Hall.

Fowler, M.: 2002; Patterns of Enterprise Application Architecture; Addison-Wesley Professional.

Fowler, M.: 2003; *Who Needs an Architect?*; **IEEE Computer Society; vol. 20(5)**: pp. 11–13.

Fricke, E. and A. P. Schulz: 2005; Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle; John Wiley and Sons Ltd.

Galal-Edeen, G. H.: 2002; *Reverse architecting: seeking the architectonic*; in Ninth Working Conference on Reverse Engineering; pp. 141–148.

Garlan, D. and B. Schmerl: 2007; *The RADAR Architecture for Personal Cognitive Assistance*; **International Journal of Software and Knowledge Engineering; vol. 17(2)**.

Ghosh, T.: 2004; *Creating Incentives for Knowledge Sharing*; *Tech. rep.*; MIT Open Courseware, Sloan school of management, Cambridge, Massachusetts, USA.

Giffin, M., O. L. Weck, G. Bounova, R. Keller, C. Eckert and P. J. Clarkson: 2007; *Change Propagation Analysis in Complex Technical Systems*; in ASME Design Engineering Technical Conferences; Las Vegas, Nevada.

Gilb, T.: 2005; Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage; Butterworth-Heinemann.

Gilb, T.: 2008; *How to Quantify Quality: Finding Scales of Measure*; **Communications in computer and Information Science; vol. 10**: pp. 27–36.

Greefhorst, D., H. Koning and H. Vliet: 2006; *The Many Faces of Architectural Descriptions*; **Information Systems Frontiers; vol. 7(3)**.

Grishman, R.: 1997; *Information Extraction: Techniques and Challenges*; **SCIE '97: International Summer School on Information Extraction**; pp. 10–27.

Group, S. W.: 2000; Systems Engineering Handbook; INCOSE.
`http://www.incose.org/ProductsPubs/products/sehandbook.aspx`

Gruba, P. and R. Al-Mahmood: 2004; *Strategies for Communication Skills Development*; in 6th Australasian Computing EducationConference (ACE2004).

Gulatti, R. K. and S. D. Eppinger: 1996; *The Coupling of Product Architecture and Organizational Structure Decisions*; *Tech. rep.*; MIT Soal School of Management.
`http://dspace.mit.edu/handle/1721.1/9839`

Haldin-Herrgard, T.: 2000; *Difficulties in Diffusion of Tacit Knowledge in Organizations*; **Journal of Intellectual Capital; vol. 1(4)**: p. 357365.

Hargis, G.: 2000; *Redeability and computer documentation*; **ACM Journal of Computer Documentation; vol. 24(3)**.

Haveman, S.: 2009; Project Buzz Tracker, Supporting System Architectures in the FWS; Master's thesis; University of Twente.

Huettel, C. A., A. W. Song and G. McCarthy: 2004; Functional magnetic resonance imaging; Sinauer Associates.

Huysman, M. and D. de Wit: 2004; *Practices of Managing Knowledge Sharing: Towards a Second Wave of Knowledge Management*; **Software Process Improvement and Practice**; **vol. 11(2)**: pp. 81–92.

IEEE: ; *ISO/IEC Std 42010 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems -Description*.
`http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html`

Isaac, D. and G. McConaughy: 1994; *The Role of Architecture and Evolvability Development in Accommodating Change*; in Proceedings of the 4th Annual Symposium of the International Council of Systems Engineering (INCOSE'94); pp. 541–545.

Jansen, A. and J. Bosch: 2005; *Software architecture as a set of architectural design decisions*; in Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA); pp. 109–119.

Jansen, A., J. van der Ven, P. Avgeriou and D. K. Hammer: 2007; *Tool support for architectural decisions*; in Proceedings of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA); pp. 44–53.

Jaring, M., R. L. Krikhaar and J. Bosch: 2004; *Representing variability in a family of MRI scanners*; **Softw. Pract. Exper.**; **vol. 34** (1): pp. 69–100.

Jauregui, J. M.: 2010; From How Much to How Many, Managing Complexity in Routine Design Automation; Ph.D. thesis; Twente University.

Kankanhalli, A., B. C. Y. Tan and K. K. Wei: 2005; *Contributing Knowledge to Electronic Knowledge Repositories: An Empirical Investigation.*; **MIS Quarterly**; **vol. 29(1)**: pp. 113–143.

Klusener, A. S., R. Lammel and C. Verhoef: 2005; *Architectural modifications to deployed software*; **Science of Computer Programming**; **vol. 54**: pp. 143–211.

Koltay, T. and S. István: 2009; *Abstracting: information literacy on a professional level*; **Journal of Documentation**; **vol. 65(5)**: pp. 841 – 855.

Koniger, P. and K. Janowitz: 1995; *Drowning in Information, But Thirsty for Knowledge*; **International Journal of Information Management**; **vol. 15(1)**: pp. 5–16.

Koning, H.: 2008; Communication of IT-Architectures; Ph.D. thesis; Universiteit Utrecht.

Koning, H., C. Dormann and H. Vliet: 2002; *Practical Guidelines for the Readeability of IT-architecture Diagrams*; in 20th Annual International Conference on computer Documentation.

Koning, H. and H. Vliet: 2006; *A Method for Defining IEEE Std 1471 Viewpoints*; **Journal of Systems and Software**; **vol. 79 (1)**: pp. 120–131.

Krikhaar, R. L.: 1997; *Reverse architecting approach for complex systems*; in IEEE International Conference on Software Maintenance; pp. 4–11; Bari.

Kwon, K. and L. F. McGinnis: 2007; *SysML-based Simulation Framework for Semiconductor Manufacturing*; in Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering.

Laar, P., S. Loo, G. Muller, T. Punter, D. Watts, P. America and J. Rutgers: 2007; *The Darwin Project: Evolvability of Software-Intensive Systems*; **23th IEEE International Conference on Software Maintenance (ICSM 2007)**.

Laar, P. d.: 2010; *Observations from the Industry-as-Laboratory Research Project Darwin*; in CSER - 8th Conference on Systems Engineering Research; pp. 658–667.

Lago, P., P. Avgeriou, R. Capilla and P. Kruchten: 2008; *Wishes and Boundaries for a Software Architecture Knowledge Community*; in 7th Working IEEE/IFIP Conference on Software Architecture (WICSA).

Lane, T. G.: 1990; *Studying Software Architecture Through Design Spaces and Rules*; *Tech. rep.*; Carnegie mellon University.

Larkin, J. H. and H. A. Simon: 1987; *Why a Diagram is (Sometimes) Worth Ten Thousand Words*; **Cognitive Science**; **vol. 11**: pp. 65–99.

Laswell, H.: 1948; The communication of ideas.; New York: Harper.

Lehman, M. M. and L. Belady: 1976; *A model of large program development*; **IBM Systems Journal**; **vol. 15(3)**: pp. 225–251.

Liang, P., A. Jansen and P. Avgeriou: 2009a; Collaborative Software Architecting through Knowledge Sharing; Springer.

Liang, P., A. Jansen and P. Avgeriou: 2009b; *Sharing architecture knowledge through models: Quality and cost*; **Knowl. Eng. Rev.**; **vol. 24**: pp. 225–244.

Liu, H., C. D. Rowles and W. X. Wen: 1995; *Critics for Knowledge-Based Design Systems*; **IEEE Transactions on knowledge and data engineering**; **vol. 7(5)**: pp. 740–750.

MacCormack, A., J. Rusnak and C. Baldwin: 2008; *The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry*; *Tech. rep.*; Hardvard Business School.
`http://hbswk.hbs.edu/item/5831.html`

MagNet: 2010; *MRI Manufacturers*.
`http://www.magnet-mri.org/resources/links/manufacturers/systems.htm`

Mayrhauser, A. V., J. Wang and Q. Li: 1999; *Experience with a reverse architecture approach to increase understanding*; in IEEE International Conference on Software Maintenance; pp. 131–138; Oxford.

McCay, B. M.: 1996; *Some Thoughts on the Quality of Computer-Based System's Architecture*; **ECBS'96**; pp. 228–234.

McRobbie, D. W., E. A. Moore, M. J. Graves and M. R. Prince: 2007; MRI From Picture to Proton; Cambridge University Press.

Mens, T., M. Wermelinger, S. Ducasse, S. Demeyer and R. Hirschfeld: 2005; *Challenges in software evolution*.

Miller, G. A.: 1956; *The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information*; **Psychological Review**; **vol. 63**: pp. 81–97.

Muller, G. J.: 2004; CAFCR: A Multi-view Method for Embedded Systems Architecting; Ph.D. thesis; Delft University of Technology.
`http://www.gaudisite.nl/index.html`

Muller, G. J.: 2006; *How to Create an Architecture Overview*.
`http://www.gaudisite.nl/OverviewHowToPaper.pdf`

Muller, G. J.: 2007a; *Coupling Enterprise and Technology by a Compact and Specific Archiecture Overview*; in INCOSE, San Diego.

Muller, G. J.: 2007b; *The Role and Task of the System Architect*.
`http://www.gaudisite.nl`

Muller, G. J.: 2008; *How reference Architectures support the evolution of Product Families*; in Proceedings of the 6th Annual Conference on Systems Engineering Research.

Muller, G. J.: 2010a; *Industry-as-Laboratory Applied in Practice: The Boderc Project*.
`http://www.gaudisite.nl/IndustryAsLaboratoryAppliedPaper.pdf`

Muller, G. J.: 2010b; *Positioning Architecting Methods in the business*.
`http://www.gaudisite.nl/PositioningCAFCRinBusinessPaper.pdf`

Muller, H. A.: 1996; *Understanding software systems using reverse engineering technologies research and practice*; in Tutorial 18th ICSE; Berlin.

Narayanan: 1997; *Diagrammatic Communication: A Taxonomy Overview*; *Tech. rep.*; CSE97-06.

Nonaka, I. and H. Takeuchi: 1995; *The Knowledge-Creating Company*; *Tech. rep.*; Oxford University Press.

OMG: ; *SysML Specification v 1.1 and Systems Modeling Language (SysML)*.

Ottino, J. M.: 2003; *Complex Systems*; **AIChE**; **vol. 49(2)**: p. 292.

Parizi, R. M. and A. A. A. Ghani: 2008; *Architectural Knowledge Sharing (AKS) Approaches: A Survey Research*; **Journal of Theoretical and Applied Information Tehcnology**.

Parsons, J.: 2002; *Effects of Local Versus Global Schema Diagrams on Verification and Communication in Conceptual Data Modeling*; **Journal of Management Information Systems**; **vol. 19(3)**: pp. 155–183.

Percivall, G.: 1994; *System Architecture for Evolutionary System Development*.

Philips: 2009; *Annual report 2009*.
`http://www.newscenter.philips.com/main/confidential/resources/corporate/quarterly_results/R_4Q09.pdf`

Poppendieck, M. and T. Poppendieck: 2003; Lean Software Development: An Agile Toolkit; Addison-Wesley Professional.

Rechtin, E. and R. W. Maier: 2000; The Art of Systems Architecting; CRC.

Richards, M. G., D. E. Hastings, A. M. Ross and D. H. Rhodes: 2007; *Design Principles for Survivable System Architecture*; **Systems Conference, 2007 1st Annual IEEE**.

Ring, J. and E. Fricke: 1998; *Rapid Evolution of All Your Systems - Problem or Opportunity?*; **Proceedings of IEEE 17th DASC, Seattle**.

Roche, S. M.: 1979; *The Thought Process in McKinsey Reports and Presentations*; *Tech. rep.*; KcKinsey.

Rowe, D. and J. Leaney: 1997; *Evaluating evolvability of computer based systems architectures - an ontological approach*; **IEEE Proc ECBS**; pp. 360–367.

Rowe, D. and J. Leaney: 1998; *Defining systems evolvability - a taxonomy for change*; **Proceedings of the International Conference and Workshop: Engineering of Computer-Based Systems (ECBS '98), Jerusalen, Israel**; pp. 45–52.

Ruben, B. D.: 1984; Communication and human behavior; Hew York: Macmillan Publishing Co.

Schilling, M. A.: 2000; *Towards a general modular systems theory and its application to inter-firm product modularity*; **Academy of Management Review**; **vol. 25**: pp. 312–334.

Schramm, W.: 1954; The process and effects of mass communication.; Urbana, IL: University of Illinois Press.

Schulz, A. P. and E. Fricke: 1999; *Incorporating flexibility, agility, robustness, and adaptability within the design of integrates systems - key to success?*; **Proceedings of IEEE/AIAA 18th Digital Avionics Syst Conf, St. Louis**.

Schulz, A. P., E. Fricke and E. Igenbergs: 2000; *Enabling Changes in Systems throughout the Entire Life-Cycle - Key to Success ?*; in Proceedings of the 10th Annual Symposium of the International Council of Systems Engineering (INCOSE'00); Minneapolis, USA.

Shannon, C. E. and W. Weaver: 1949; The Mathematical Theory of Communication; The University of Illinois Press, Urbana, IL.

Shaw, M.: 1989; *Larger scale systems require higher-level abstractions*; **Proceedings of the 5th International Workshop on Software Specification and Design**.

Simon, H. A.: 1962; *The Architecture of Complexity*; in American Philosophical Society; vol. 106; pp. 467–482–; MIT Press, Cambridge.

Simpson, W. and L. Prusak: 1995; *Troubles with information overload - moving drom quality to quantity in information provision*; **International Journal of Information Management**; **vol. 15(6)**: pp. 413–425.

Smolander, K. and T. Paivarinta: 2002; *Practical Rationale for Describing Software Architecture, Beyond Programming-in-the-Large*; in IEEE/IFIP Conference on Software Architecture (WICSA3).

Sobek II, D. K. and C. Jimmerson: 2004; *A3 Reports: Tool for Process Improvement*; in Industrial Engineering Research Conference; pp. –; Montana State University, Houston.

Sobek II, D. K. and A. Smalley: 2008; Understanding A3 Thinking, A Critical Component of Toyota's PDCA Management System; CRC Press.

Spence, R.: 2001; Information Visualization; Addison-Wesley.

Steiner, R.: 1998; *Systems Architecture and Evolvability - Definitions and Perspective*; **Proceedings of the 8th Annual Symposium of the International Council on System Engineering**.

Tang, A., M. A. Babar, I. Gorton and J. Han: 2006; *A survey of architecture design rationale*; **The Journal of Systems & Software**; **vol. 79(12)**: pp. 1792–1804.

Taylor, S. E. and S. T. Fiske: 1975; *Point of view and perceptions of causality*; **Journal of Personality and Social Psycology**; **vol. 32(3)**: pp. 439–445.

Theodorson, S. and A. Theodorson: 1969; A modern dictionary of sociology; New York: Cassell Education Limited.

Tory, M. and T. Moller: 2004; *Human Factors in Visualization Research*; **IEEE Transactions on Visualization and Computer Graphics**; **vol. 10**: pp. 72–84.

Tufte, E. R.: 1990; Envisioning Information; Graphics Press.

Tufte, E. R.: 1997; Visual Explanations: Images and Quantities, Evidence and Narrative; Graphics Press.

Tufte, E. R.: 2001; The Visual Display of Quantitative Information; Graphics Press.

Van Vliet, H.: 2000; Software Engineering: Principles and Practice; John Wiley & Sons.

Verdü, S.: 2000; Fifty years of Shannon theory; IEEE Press.

Wagner, G. P. and L. Altenberg: 1996; *Complex adaptations and the evolution of evolvability*; **Evolution 50**; pp. 967–976.

Wang, R. and C. Dagli: 2008; *An Executable System Architecture Approach to Discrete Events System Modeling Using SysML in Conjunction with Colored Petri Net*; in SysCom 2008, IEEE International Systems Conference.

Ware, C.: 2000; Information Visualization; Morgan Kaufmann.

Weishaupt, D., V. D. Koechli and B. Marincek: 2006; How does MRI work?: An Introduction to the Physics and Function of Magnetic Resonance Imaging; Springer.

Wikipedia: 2010; *Modeling language*.
`http://en.wikipedia.org/wiki/Modeling_language`

Williams, L. J.: 1982; *Congnitive Load and the Functional Field of View*; **The Human Factors Society**; **vol. 24**: pp. 683–692.

Womack, J., D. T. Jones and D. C. Ross: 1990; The Machine that Changed the World: The Story of Lean Production; Free Press, New York.

Woods, D. D. and E. Hollnagel: 2005; Joint Cognitive Systems: Foundations of Cognitive Systems Engineering.; Taylor & Francis.

Yourdon, E.: 1989; Modern Structured Analysis; Prentice Hall.

Zachman, J.: 1987; *A framework for information systems architecture*; **IBM Systems Journal**; **vol. 26:3**.

Appendices

Appendix A

Surveys

The main purpose of the surveys is to collect insight from stakeholders. They are not meant to collect hard evidence but to provide insight about the voice of the customer. In this context our main customers are architects, and those stakeholders that have to communicate with them.

Surveys gather data about different key aspects for this Thesis. Among those aspects are; product development barriers, effectiveness of current ways to capture and share architectural knowledge (e.g. SDS), what architects need to share architectural knowledge, challenges to communicate architectural knowledge, effectiveness of the A3 Architecture Overviews as a communication tool, effort to the creation of A3 Architecture Overviews, and a few other aspects. Three dedicated surveys were developed to collect the desired data. A survey example is shown in Figure A.1.

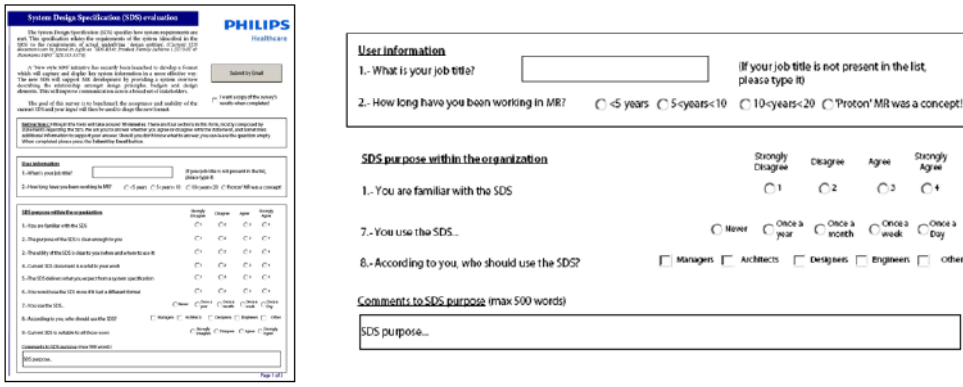


Figure A.1: Survey example

As shown in Figure A.1, in most cases the surveyed were asked to fill in whether he/she *strongly agree*, *agree*, *disagree* or *strongly disagree* with the statement presented. Statements that were left empty were considered "don't know" answers. In the survey, some questions requested specific quantification. There was room in the survey to collect additional comments and feedback. The process to collect the data was as follows:

1. Select targets: Depending on the goal of the survey, a list of employees was created.

2. Populate survey: Email was chosen as the channel to populate the surveys. Selected employees were addressed through email, with an introductory text explaining the goal of the survey, and the electronic survey.

3. Collect data: Once the target filled in the survey, the survey automatically delivered the data to a predefined email address.

4. Analyze data: After a predefined waiting time, data collected was analyzed. As shown in Figure A.2, an Excel application was developed to analyze and present the data. As shown in Figure A.2, job title and working experience was taken into account during the analysis.

5. Create report: A report with the findings and data gathered was created and populated to key stakeholders and surveyed that explicitly requested it in the survey.

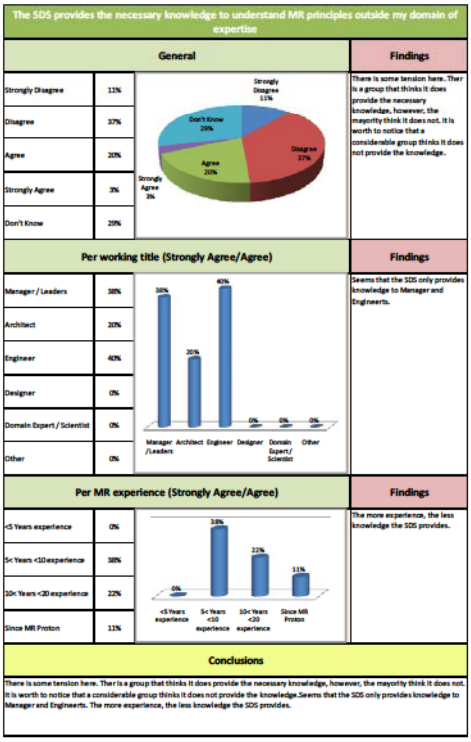


Figure A.2: Question Analysis

A.1 Survey I: Development Concerns and System Design Specification Evaluation

The goal of this survey was to understand major development challenges, effectiveness of vital documents such as the SDS as a way to capture design knowledge, and collect feedback from practitioners. The survey had around 40 questions divided in four different sections.

The target of this survey was the MRI development organization (approximately 250 employees). Marketing, sales and related fields were not addressed. After 4 weeks 35 employees replied to the survey request (around 1/7 of MRI development population).

A.1.1 Introductory Text

System Design Specification (SDS) evaluation

The **System Design Specification** (SDS) specifies how system requirements are met. This specification relates the requirements of the system (described in the SRS) to the requirements of actual underlying 'design entities'. (Current SDS document can be found in Agile as "SDS R2.6.3" XJS351-02762).

A ***'New style SDS'*** initiative has recently been launched to develop a format which will capture and display key system information in a more effective way. The new SDS will support MR development by providing a system overview describing the relationship amongst design principles, budgets and design elements. This will improve communication across a broad set of stakeholders.

The goal of this survey is to benchmark the acceptance and usability of the current SDS and your input will then be used to shape the new format.

Instructions: Filling in the form will take around 10 minutes. There are four sections in this form, mostly composed by statements regarding the SDS. We ask you to answer whether you agree or disagree with the statement, and sometimes additional information to support your answer. Should you don't know what to answer, you can leave the question empty. When completed please press the **Submit by Email** button.

A.1.2 Data Analysis

Results are provided in Tables A.1, A.2, A.3, A.4, A.5, A.6, A.7. Only the most relevant questions for the goal of this Thesis are provided.

A.1.3 Threats to validity

We assume that we have enough representatives for each group to derive some conclusions from the analysis. The group that may lack representation is Domain Expert. It could be argued that this group could be aggregated to Other group. Domain Expert group was however treated as an independent group as their input was very different from the rest of the MRI population, deserving their own group.

Threat to validity: Only two Domain experts replied to the survey, therefore answers are always 0%, 50% or 100%. Although Domain Experts play a key role in the development process, there are few Domain Experts within Philips Healthcare MRI. Consequently it is difficult to collect data from this group.

A.2 Survey II: Evaluation of A3 Architecture Overviews as Communication Tool

The goal of this survey was to evaluate whether A3 Architecture Overviews are perceived by the users as an effective communication tool, especially in comparison with other approaches.

To that end, A3 Architecture Overviews of different system aspects were arranged in folders, and emailed along with and introductory text to employees that should have an interest in those system aspects. Nine A3 Architecture Overviews were provided to 52 employees. After two weeks, a survey with approximately 35 questions was sent to collect feedback from those employees that used one or more of them in their current projects. After 4 weeks, 17 employees replied to the survey.

A.2.1 Introductory Text

Improving product development through effective communication: A3 Architecture Overviews

Effective communication is a challenge for product development in most companies. A study realized over 140 companies by the Aberdeen Group showed communication as the biggest problem companies face. It also showed that 71% of the leading companies had 'improving communication' as the primary target to improve their development process.

An internal survey at Philips Healthcare MR (see attached document) showed similar results. The main barriers we face are: **Communication across disciplines and departments** (71%), **lack of knowledge sharing** (61%), **managing system complexity** (77%), **lack of system overview** (74%), and **finding system information** (57%). Those barriers were identified as the root cause of many development problems and bad decisions.

To improve communication and therefore improve our development process, the approach ***A3 Architecture Overviews*** has been developed by Daniel Borches in the context of the Darwin project. An A3 Architecture Overview provides a model view of a system aspect on one side of an A3 sheet, while the other side displays a textual view to support and complement the model view. The principle behind the A3 is to include relevant information in an structured way to create a complete picture of the topic at hand, and eliminate everything else until only the essentials remain.

We provide you with 4 sets of A3 Architecture Overviews to help you with some MRI aspects: **DDAS network architecture, MRI Scan Control, Functional Clusters** and **Calibrations** (see details below). We ask you to choose the set or sets that are most attractive and/or interesting to you (after a short scan) and read them. Please limit the time you spend reading the selected A3s. After two weeks, when you have had time to read them, we kindly ask you to help us evaluate the effectiveness of the approach by filling in a short survey.

- *DDAS network architecture*: DDAS is the evolution of current CDAS (Control & Data Acquisition) System. This project will bring a number of changes and opportunities to the MRI product and the development organization.

- *MRI Scan Control*: The MRI System requires subsystems to be precisely synchronized and to be able to execute actions timely. Scan Control is responsible for this.

- *Functional Clusters*: Recently the MRI system SW has been partitioned in so-called functional clusters for which architecture overviews are being written. One of those functional clusters is Reconstruction SW.

- *Calibrations*: Calibrations are techniques to adjust measured imperfections in order to ensure proper performance of specific system qualities. An important calibration is Resonance Frequency Calibration.

A.2.2 Data Analysis

As shown in Figure A.2, feedback from users have been analyzed taking into account job profile and working experience. Results are provided in Tables A.8, A.9, A.10, A.11, A.12, A.13, A.14, A.15, A.16. Only the most relevant questions for the goal of this Thesis are provided.

A.2.3 Threats to validity

As in the previous survey, we assume that we have enough representatives for each group to derive some conclusions from the analysis. Again, the group that may lack representation is Domain Expert.

Threat to validity: Only one Domain expert replied to the survey, therefore answers are always 0% or 100%.

%clearpage

A.3 Survey III: Evaluation of A3 Architecture Overviews Creation

The goal of this survey was to evaluate the creation effort of A3 Architecture Overviews. The target of this survey were those employees who had created A3 Architecture Overviews as a part of their daily work for a protect in which they were already involved. 5 people were actively involved in the research and creating and using A3 Architecture Overviews for their own projects. They were asked to fill in a survey with 16 questions. After two weeks 4 of them replied.

A.3.1 Introductory Text

Dear A3 Architecture Overview author. In order to evaluate the effort required to create and A3 Architecture Overview, the following survey has been developed. The goal of this survey is to evaluate the creation effort and collect your experiences during the process.

A.3.2 Data Analysis

Feedback from creators of A3 Architecture Overviews have been analyzed. Results are provided in Table A.17.

A.3.3 Threats to validity

The creation of A3 Architecture Overviews by practitioners required commitment. Five people committed to create and use A3 Architecture Overviews for their current projects. Four of them replied to the survey, however due to the limited number of participants, there is probably not enough representatives to derive conclusions. However, as feedback from those participants is quite similar, it provides enough insight to provide some directions.

Statement 1: When developing the part you are responsible for, you have a system overview to support the process					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	6%	Manager/Leader	50%	<5 Years	75%
Agree	40%	Architect	80%	5 <Years< 10	31%
Disagree	34%	Engineer	30%	10 <Years< 20	56%
Strongly Disagree	6%	Designer	43%	Since MR Proton	44%
Don't Know	14%	Domain Expert	100%	(> 20 Years)	
		Other	0%		
Statement 2: Having a system overview supports you in your development activities					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	23%	Manager/Leader	75%	<5 Years	75%
Agree	51%	Architect	100%	5 <Years< 10	77%
Disagree	9%	Engineer	70%	10 <Years< 20	78%
Strongly Disagree	0%	Designer	71%	Since MR Proton	68%
Don't Know	17%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 3: You can easily find the system information you need to cope with your work					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	3%	Manager/Leader	38%	<5 Years	50%
Agree	20%	Architect	20%	5 <Years< 10	38%
Disagree	49%	Engineer	30%	10 <Years< 20	0%
Strongly Disagree	9%	Designer	0%	Since MR Proton	11%
Don't Know	20%	Domain Expert	50%	(> 20 Years)	
		Other	0%		
Statement 4: Requirements are clearly expressed and transformed into design specifications					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	0%	<5 Years	25%
Agree	17%	Architect	40%	5 <Years< 10	15%
Disagree	46%	Engineer	30%	10 <Years< 20	11%
Strongly Disagree	20%	Designer	0%	Since MR Proton	22%
Don't Know	17%	Domain Expert	50%	(> 20 Years)	
		Other	0%		
Statement 5: The interfaces between elements and between subsystems are clear					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	25%	<5 Years	25%
Agree	17%	Architect	20%	5 <Years< 10	8%
Disagree	49%	Engineer	10%	10 <Years< 20	22%
Strongly Disagree	6%	Designer	29%	Since MR Proton	22%
Don't Know	29%	Domain Expert	0%	(> 20 Years)	
		Other	0%		
Statement 6: You are familiar with most design principles and techniques (e.g. calibrations)					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	11%	Manager/Leader	38%	<5 Years	25%
Agree	37%	Architect	100%	5 <Years< 10	38%
Disagree	26%	Engineer	60%	10 <Years< 20	44%
Strongly Disagree	3%	Designer	14%	Since MR Proton	78%
Don't Know	23%	Domain Expert	50%	(> 20 Years)	
		Other	33%		
Statement 7: The relation between design principles and derived budgets is clear to you					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	3%	Manager/Leader	38%	<5 Years	25%
Agree	34%	Architect	100%	5 <Years< 10	23%
Disagree	31%	Engineer	20%	10 <Years< 20	44%
Strongly Disagree	9%	Designer	14%	Since MR Proton	56%
Don't Know	23%	Domain Expert	50%	(> 20 Years)	
		Other	33%		

Table A.1: Development Challenges I

Statement 1: A text-based document is preferred to a model-based description for the system design					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	9%	Manager/Leader	50%	<5 Years	50%
Agree	26%	Architect	20%	5 <Years< 10	23%
Disagree	23%	Engineer	30%	10 <Years< 20	44%
Strongly Disagree	14%	Designer	14%	Since MR Proton	33%
Don't Know	29%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 2: The size of a document is not an issue for its use (e.g. number of pages)					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	11%	Manager/Leader	25%	<5 Years	25%
Agree	40%	Architect	40%	5 <Years< 10	62%
Disagree	29%	Engineer	80%	10 <Years< 20	78%
Strongly Disagree	0%	Designer	43%	Since MR Proton	22%
Don't Know	20%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 3: Semantics and domain-specific jargon in documents is not a problem					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	9%	Manager/Leader	50%	<5 Years	50%
Agree	43%	Architect	80%	5 <Years< 10	69%
Disagree	20%	Engineer	60%	10 <Years< 20	44%
Strongly Disagree	0%	Designer	43%	Since MR Proton	33%
Don't Know	29%	Domain Expert	50%	(> 20 Years)	
		Other	0%		
Statement 4: You have a way (e.g. method, 'tool') that supports you when making design decisions					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	25%	<5 Years	50%
Agree	34%	Architect	40%	5 <Years< 10	31%
Disagree	34%	Engineer	30%	10 <Years< 20	33%
Strongly Disagree	6%	Designer	57%	Since MR Proton	33%
Don't Know	26%	Domain Expert	50%	(> 20 Years)	
		Other	0%		
Statement 5: Communication across disciplines is a problem that affects your work					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	23%	Manager/Leader	63%	<5 Years	75%
Agree	46%	Architect	80%	5 <Years< 10	62%
Disagree	14%	Engineer	70%	10 <Years< 20	89%
Strongly Disagree	0%	Designer	57%	Since MR Proton	56%
Don't Know	17%	Domain Expert	100%	(> 20 Years)	
		Other	67%		
Statement 6: Communication across departments is a problem that affects your work					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	29%	Manager/Leader	63%	<5 Years	75%
Agree	43%	Architect	80%	5 <Years< 10	69%
Disagree	9%	Engineer	80%	10 <Years< 20	89%
Strongly Disagree	0%	Designer	57%	Since MR Proton	56%
Don't Know	20%	Domain Expert	100%	(> 20 Years)	
		Other	67%		
Statement 7: MR system complexity is a problem when dealing with new developments					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	14%	Manager/Leader	75%	<5 Years	75%
Agree	63%	Architect	100%	5 <Years< 10	85%
Disagree	9%	Engineer	60%	10 <Years< 20	78%
Strongly Disagree	0%	Designer	86%	Since MR Proton	67%
Don't Know	14%	Domain Expert	100%	(> 20 Years)	
		Other	67%		

Table A.2: Development Challenges II

Statement 1: You are familiar with the SDS					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	20%	Manager/Leader	63%	<5 Years	50%
Agree	29%	Architect	80%	5 <Years< 10	23%
Disagree	34%	Engineer	30%	10 <Years< 20	67%
Strongly Disagree	17%	Designer	14%	Since MR Proton	67%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	
		Other	67%		
Statement 2: The purpose of the SDS is clear to you					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	34%	Manager/Leader	100%	<5 Years	100%
Agree	43%	Architect	100%	5 <Years< 10	54%
Disagree	23%	Engineer	60%	10 <Years< 20	78%
Strongly Disagree	0%	Designer	57%	Since MR Proton	100%
Don't Know	0%	Domain Expert	50%	(> 20 Years)	
		Other	100%		
Statement 3: The utility of the SDS (when and where to use it) is clear to you					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	14%	Manager/Leader	100%	<5 Years	75%
Agree	49%	Architect	100%	5 <Years< 10	46%
Disagree	34%	Engineer	50%	10 <Years< 20	67%
Strongly Disagree	0%	Designer	14%	Since MR Proton	78%
Don't Know	3%	Domain Expert	50%	(> 20 Years)	
		Other	67%		
Statement 4: Current SDS document is useful for your work					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	50%	<5 Years	75%
Agree	29%	Architect	40%	5 <Years< 10	23%
Disagree	40%	Engineer	30%	10 <Years< 20	22%
Strongly Disagree	14%	Designer	0%	Since MR Proton	22%
Don't Know	17%	Domain Expert	50%	(> 20 Years)	
		Other	0%		
Statement 5: The SDS delivers what you expect from a system specification					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	25%	<5 Years	50%
Agree	26%	Architect	20%	5 <Years< 10	31%
Disagree	49%	Engineer	40%	10 <Years< 20	11%
Strongly Disagree	6%	Designer	0%	Since MR Proton	22%
Don't Know	20%	Domain Expert	50%	(> 20 Years)	
		Other	33%		
Statement 6: You would use the SDS more if it had a different format					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	6%	Manager/Leader	38%	<5 Years	25%
Agree	40%	Architect	80%	5 <Years< 10	23%
Disagree	29%	Engineer	30%	10 <Years< 20	67%
Strongly Disagree	3%	Designer	43%	Since MR Proton	67%
Don't Know	23%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 7: The SDS provides you with the overview you need					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	75%	<5 Years	50%
Agree	43%	Architect	20%	5 <Years< 10	46%
Disagree	40%	Engineer	40%	10 <Years< 20	44%
Strongly Disagree	10%	Designer	43%	Since MR Proton	33%
Don't Know	17%	Domain Expert	50%	(> 20 Years)	
		Other	0%		

Table A.3: System Design Specification (SDS) Evaluation I

Statement 1: The SDS provides you sufficient information to estimate the impact of a change in the system					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	13%	<5 Years	8%
Agree	14%	Architect	0%	5 <Years< 10	23%
Disagree	57%	Engineer	40%	10 <Years< 20	11%
Strongly Disagree	9%	Designer	0%	Since MR Proton	0%
Don't Know	20%	Domain Expert	0%	(> 20 Years)	
		Other	0%		
Statement 2: The budget (e.g. power budget) you have for a system aspect and the rationale is not clear					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	9%	Manager/Leader	50%	<5 Years	50%
Agree	43%	Architect	80%	5 <Years< 10	46%
Disagree	17%	Engineer	30%	10 <Years< 20	56%
Strongly Disagree	0%	Designer	71%	Since MR Proton	56%
Don't Know	31%	Domain Expert	100%	(> 20 Years)	
		Other	0%		
Statement 3: When looking for insight on the system, there are better alternatives than the SDS					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	13%	<5 Years	25%
Agree	40%	Architect	80%	5 <Years< 10	38%
Disagree	31%	Engineer	30%	10 <Years< 20	56%
Strongly Disagree	3%	Designer	43%	Since MR Proton	33%
Don't Know	26%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 4: The information provided by the SDS is reliable					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	38%	<5 Years	50%
Agree	40%	Architect	40%	5 <Years< 10	23%
Disagree	31%	Engineer	40%	10 <Years< 20	56%
Strongly Disagree	3%	Designer	29%	Since MR Proton	44%
Don't Know	26%	Domain Expert	50%	(> 20 Years)	
		Other	67%		
Statement 5: The use of the SDS should be encouraged within the organization					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	23%	Manager/Leader	88%	<5 Years	75%
Agree	63%	Architect	100%	5 <Years< 10	92%
Disagree	0%	Engineer	90%	10 <Years< 20	89%
Strongly Disagree	0%	Designer	71%	Since MR Proton	78%
Don't Know	14%	Domain Expert	100%	(> 20 Years)	
		Other	67%		
Statement 6: The system partition used in the SDS is appropriate to your needs					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	6%	Manager/Leader	38%	<5 Years	0%
Agree	43%	Architect	40%	5 <Years< 10	69%
Disagree	26%	Engineer	60%	10 <Years< 20	56%
Strongly Disagree	0%	Designer	57%	Since MR Proton	33%
Don't Know	26%	Domain Expert	100%	(> 20 Years)	
		Other	0%		
Statement 7: It is not easy to find the information in the SDS document					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	3%	Manager/Leader	63%	<5 Years	50%
Agree	49%	Architect	60%	5 <Years< 10	46%
Disagree	23%	Engineer	40%	10 <Years< 20	67%
Strongly Disagree	0%	Designer	57%	Since MR Proton	44%
Don't Know	26%	Domain Expert	50%	(> 20 Years)	
		Other	33%		

Table A.4: System Design Specification (SDS) Evaluation II

Statement 1: The level of detail used in the SDS is appropriate					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	50%	<5 Years	25%
Agree	43%	Architect	40%	5 <Years< 10	46%
Disagree	26%	Engineer	60%	10 <Years< 20	44%
Strongly Disagree	3%	Designer	0%	Since MR Proton	44%
Don't Know	29%	Domain Expert	100%	(> 20 Years)	
		Other	33%		
Statement 2: Formal notation (e.g. UML) should be used in the SDS					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	3%	Manager/Leader	0%	<5 Years	0%
Agree	26%	Architect	40%	5 <Years< 10	38%
Disagree	37%	Engineer	50%	10 <Years< 20	44%
Strongly Disagree	6%	Designer	43%	Since MR Proton	11%
Don't Know	29%	Domain Expert	0%	(> 20 Years)	
		Other	0%		
Statement 3: You do not mind a different format for the SDS (e.g. A3 instead of A4)					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	6%	Manager/Leader	50%	<5 Years	0%
Agree	43%	Architect	60%	5 <Years< 10	69%
Disagree	31%	Engineer	60%	10 <Years< 20	22%
Strongly Disagree	0%	Designer	43%	Since MR Proton	67%
Don't Know	20%	Domain Expert	50%	(> 20 Years)	
		Other	0%		

Table A.5: System Design Specification (SDS) Evaluation III

Please rate the readability (easy to read) of the SDS					
Very bad	0%	Bad	17%	Neutral	20%
Good	40%	Very good	0%	Don't know	23%
Please rate the understandability (easy to understand) of the SDS					
Very bad	0%	Bad	6%	Neutral	34%
Good	37%	Very good	0%	Don't know	23%
Please rate the usability (easy to use) of the SDS					
Very bad	0%	Bad	43%	Neutral	20%
Good	14%	Very good	0%	Don't know	23%
Please rate the accuracy of information of the SDS					
Very bad	0%	Bad	11%	Neutral	40%
Good	20%	Very good	0%	Don't know	29%

Table A.6: System Design Specification (SDS) Characteristics

Question: Have you or your team had situations in which having more knowledge of a specific design principle would have prevented a problem or helped to make a better decision?					
Never	14%	Once a year	37%	Once a month	23%
Once a week	0%	Once a day	0%	Don't know	26%

Table A.7: Lack of Knowledge Sharing Impact

Statement 1: Reading documents outside my domain of expertise is sometimes a challenge					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	20%	Manager/Leader	100%	<5 Years	100%
Agree	60%	Architect	100%	5 <Years< 10	50%
Disagree	13%	Engineer	50%	10 <Years< 20	75%
Strongly Disagree	0%	Designer	67%	Since MR Proton	100%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 2: Communication with other disciplines or departments requires a common model					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	47%	Manager/Leader	100%	<5 Years	100%
Agree	40%	Architect	100%	5 <Years< 10	100%
Disagree	7%	Engineer	100%	10 <Years< 20	75%
Strongly Disagree	0%	Designer	33%	Since MR Proton	75%
Don't Know	7%	Domain Expert	100%		
Statement 3: For communication, an overview of a specific topic is better than plenty of information					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	60%	Manager/Leader	100%	<5 Years	100%
Agree	33%	Architect	100%	5 <Years< 10	100%
Disagree	0%	Engineer	100%	10 <Years< 20	75%
Strongly Disagree	0%	Designer	67%	Since MR Proton	100%
Don't Know	7%	Domain Expert	100%	(> 20 Years)	
Statement 4: Finding time to read available documentation about a topic is not a problem					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	0%	<5 Years	0%
Agree	7%	Architect	17%	5 <Years< 10	0%
Disagree	53%	Engineer	0%	10 <Years< 20	25%
Strongly Disagree	33%	Designer	0%	Since MR Proton	0%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 5: Documents are useful during discussions					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	13%	Manager/Leader	67%	<5 Years	67%
Agree	40%	Architect	50%	5 <Years< 10	75%
Disagree	40%	Engineer	50%	10 <Years< 20	25%
Strongly Disagree	0%	Designer	33%	Since MR Proton	50%
Don't Know	7%	Domain Expert	100%	(> 20 Years)	
Statement 6: In my meetings there is usually a model to support the discussion					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	33%	<5 Years	33%
Agree	20%	Architect	17%	5 <Years< 10	0%
Disagree	60%	Engineer	0%	10 <Years< 20	0%
Strongly Disagree	13%	Designer	33%	Since MR Proton	50%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 7: Insight obtained at meetings is captured effectively					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	0%	<5 Years	0%
Agree	7%	Architect	0%	5 <Years< 10	0%
Disagree	67%	Engineer	0%	10 <Years< 20	0%
Strongly Disagree	20%	Designer	33%	Since MR Proton	25%
Don't Know	10%	Domain Expert	0%	(> 20 Years)	
Statement 8: Usually I do not have the right information to make a (correct) decision					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	13%	Manager/Leader	0%	<5 Years	0%
Agree	20%	Architect	50%	5 <Years< 10	25%
Disagree	53%	Engineer	50%	10 <Years< 20	50%
Strongly Disagree	7%	Designer	0%	Since MR Proton	50%
Don't Know	10%	Domain Expert	100%	(> 20 Years)	

Table A.8: A3 Architecture Overview: Communication Challenges I

Statement 1: I am overloaded with information...					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	100%	<5 Years	67%
Agree	33%	Architect	67%	5 <Years< 10	50%
Disagree	33%	Engineer	50%	10 <Years< 20	50%
Strongly Disagree	0%	Designer	33%	Since MR Proton	75%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 2: ...yet I do not find the information I need					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	7%	Manager/Leader	100%	<5 Years	100%
Agree	60%	Architect	67%	5 <Years< 10	75%
Disagree	27%	Engineer	100%	10 <Years< 20	50%
Strongly Disagree	0%	Designer	33%	Since MR Proton	50%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 3: There are not enough design review meetings					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	13%	Manager/Leader	67%	<5 Years	33%
Agree	33%	Architect	67%	5 <Years< 10	50%
Disagree	40%	Engineer	50%	10 <Years< 20	50%
Strongly Disagree	0%	Designer	0%	Since MR Proton	50%
Don't Know	13%	Domain Expert	0%	(> 20 Years)	
Statement 4: It is hard to visualize the impact of a local change in the system					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	67%	<5 Years	67%
Agree	33%	Architect	83%	5 <Years< 10	50%
Disagree	27%	Engineer	50%	10 <Years< 20	75%
Strongly Disagree	0%	Designer	33%	Since MR Proton	50%
Don't Know	13%	Domain Expert	0%	(> 20 Years)	

Table A.9: A3 Architecture Overview: Communication Challenges II

Please rate the readability (easy to read) of current documents you use					
Very bad	0%	Bad	27%	Neutral	40%
Good	27%	Very good	0%	Don't know	7%
Please rate the understandability (easy to understand) of current documents you use					
Very bad	0%	Bad	20%	Neutral	47%
Good	27%	Very good	0%	Don't know	7%
Please rate the usability (easy to use e.g. at meetings) of current documents you use					
Very bad	0%	Bad	47%	Neutral	33%
Good	7%	Very good	7%	Don't know	7%
Please rate the adequate amount of information of current documents you use					
Very bad	0%	Bad	20%	Neutral	40%
Good	33%	Very good	0%	Don't know	7%

Table A.10: Current Documentation Used: Characteristics

Statement 1: The notation used in the A3s is complicated					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	0%	Manager/Leader	0%	<5 Years	0%
Agree	20%	Architect	50%	5 <Years< 10	25%
Disagree	67%	Engineer	0%	10 <Years< 20	25%
Strongly Disagree	7%	Designer	0%	Since MR Proton	25%
Don't Know	7%	Domain Expert	0%	(> 20 Years)	
Statement 2: Having the same layout / structure among A3 Architecture Overviews makes it easier to read other A3					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	20%	Manager/Leader	100%	<5 Years	75%
Agree	60%	Architect	83%	5 <Years< 10	100%
Disagree	13%	Engineer	50%	10 <Years< 20	50%
Strongly Disagree	0%	Designer	67%	Since MR Proton	75%
Don't Know	7%	Domain Expert	100%	(> 20 Years)	

Table A.11: A3 Architecture Overview: Contents & Structure

Please, rate how useful the following elements of the A3 Architecture Overview are:					
Functional View					
Useless	0%	Little Use	0%	Neutral	13%
Useful	53%	Very Useful	27%	Don't know	7%
Visual Aids					
Useless	0%	Little Use	0%	Neutral	0%
Useful	60%	Very Useful	33%	Don't know	7%
Quantification View					
Useless	0%	Little Use	13%	Neutral	27%
Useful	47%	Very Useful	7%	Don't know	7%
Physical View					
Useless	0%	Little Use	7%	Neutral	7%
Useful	33%	Very Useful	47%	Don't know	7%
Annotations (Design Constraints)					
Useless	0%	Little Use	0%	Neutral	40%
Useful	47%	Very Useful	7%	Don't know	7%
Color					
Useless	0%	Little Use	7%	Neutral	13%
Useful	73%	Very Useful	0%	Don't know	7%
Legend					
Useless	0%	Little Use	0%	Neutral	27%
Useful	53%	Very Useful	13%	Don't know	7%
A3 Summary (Text View)					
Useless	0%	Little Use	7%	Neutral	13%
Useful	73%	Very Useful	0%	Don't know	7%
Links within Views (e.g. numbers)					
Useless	0%	Little Use	0%	Neutral	27%
Useful	60%	Very Useful	7%	Don't know	7%

Table A.12: A3 Architecture Overview: Elements

Statement 1: The A3 size is a problem for its use					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	7%	Manager/Leader	0%	<5 Years	0%
Agree	13%	Architect	33%	5 <Years< 10	0%
Disagree	80%	Engineer	0%	10 <Years< 20	25%
Strongly Disagree	7%	Designer	0%	Since MR Proton	25%
Don't Know	0%	Domain Expert	0%	(> 20 Years)	
Statement 2: I like the idea of having different views within the same A3					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	100%	<5 Years	100%
Agree	73%	Architect	100%	5 <Years< 10	100%
Disagree	0%	Engineer	100%	10 <Years< 20	100%
Strongly Disagree	0%	Designer	100%	Since MR Proton	100%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	
Statement 3: To learn a new topic, I think a regular document is as good as an A3 Architecture Overview					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	13%	Manager/Leader	0%	<5 Years	33%
Agree	13%	Architect	17%	5 <Years< 10	50%
Disagree	60%	Engineer	0%	10 <Years< 20	0%
Strongly Disagree	13%	Designer	67%	Since MR Proton	25%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	
Statement 4: I don't see the value of having the information within an A3 instead of on several A4 pages					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	7%	Manager/Leader	0%	<5 Years	0%
Agree	7%	Architect	33%	5 <Years< 10	25%
Disagree	60%	Engineer	0%	10 <Years< 20	0%
Strongly Disagree	27%	Designer	0%	Since MR Proton	25%
Don't Know	0%	Domain Expert	0%	(> 20 Years)	
Statement 5: I would like an A3 Architecture Overview of my current work					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	7%	Manager/Leader	33%	<5 Years	33%
Agree	73%	Architect	83%	5 <Years< 10	75%
Disagree	7%	Engineer	100%	10 <Years< 20	100%
Strongly Disagree	7%	Designer	100%	Since MR Proton	100%
Don't Know	7%	Domain Expert	100%	(> 20 Years)	
Statement 6: I would use the A3 Architecture Overview as a discussion tool (e.g. at meetings)					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	100%	<5 Years	100%
Agree	60%	Architect	100%	5 <Years< 10	75%
Disagree	13%	Engineer	100%	10 <Years< 20	100%
Strongly Disagree	0%	Designer	67%	Since MR Proton	75%
Don't Know	0%	Domain Expert	0%	(> 20 Years)	
Statement 7: I would use the A3 Architecture Overview as a collaboration tool (e.g. share knowledge among disciplines)					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	100%	<5 Years	100%
Agree	73%	Architect	100%	5 <Years< 10	100%
Disagree	0%	Engineer	100%	10 <Years< 20	100%
Strongly Disagree	0%	Designer	100%	Since MR Proton	100%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	
Statement 8: I rather read a document than an A3 Architecture Overview					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	7%	Manager/Leader	0%	<5 Years	33%
Agree	13%	Architect	17%	5 <Years< 10	25%
Disagree	60%	Engineer	50%	10 <Years< 20	25%
Strongly Disagree	20%	Designer	33%	Since MR Proton	0%
Don't Know	0%	Domain Expert	0%	(> 20 Years)	

Table A.13: A3 Architecture Overview: Communication Tool I

Statement 1: I would like to have more A3 Architecture Overviews of other system aspects					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	27%	Manager/Leader	100%	<5 Years	67%
Agree	60%	Architect	83%	5 <Years< 10	75%
Disagree	13%	Engineer	100%	10 <Years< 20	100%
Strongly Disagree	0%	Designer	67%	Since MR Proton	100%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	
Statement 2: A3 Architecture Overviews can be used to estimate and communicate the impact of a change in the system					
General Response		*Strongly Agree/Agree per Job Title*		*Strongly Agree/Agree per Experience*	
Strongly Agree	13%	Manager/Leader	33%	<5 Years	67%
Agree	73%	Architect	100%	5 <Years< 10	75%
Disagree	13%	Engineer	50%	10 <Years< 20	75%
Strongly Disagree	0%	Designer	100%	Since MR Proton	100%
Don't Know	0%	Domain Expert	100%	(> 20 Years)	

Table A.14: A3 Architecture Overview: Communication Tool II

Please rate the readability (easy to read) of A3 Architecture Overviews you used					
Very bad	0%	Bad	0%	Neutral	27%
Good	53%	Very good	20%	Don't know	0%
Please rate the understandability (easy to understand) of A3 Architecture Overviews you used					
Very bad	0%	Bad	7%	Neutral	33%
Good	53%	Very good	7%	Don't know	0%
Please rate the usability (easy to use e.g. at meetings) of A3 Architecture Overviews you used					
Very bad	0%	Bad	0%	Neutral	13%
Good	73%	Very good	13%	Don't know	0%
Please rate the adequate amount of information of A3 Architecture Overviews you used					
Very bad	0%	Bad	0%	Neutral	27%
Good	67%	Very good	7%	Don't know	0%

Table A.15: A3 Architecture Overview: Characteristics

Strong Points for Users
- Gives very good insight in the impact of the system (also for managers). - Good communication tool. - It does indeed provide overview. - The value of the approach is expected to be strongly amplified when used by everyone. - The lack of drawings in current documentation and during meetings to communicate is the main motivation for adopting this approach. - It is very useful in meetings. - It is useful to have a compact way to present information.
Weak Points for Users
- Still very much information. - In occasions it requires explanation by an expert to fully understand the concept. - There is some clutter in the overviews (e.g. color) - The use of color is incompatible with Black&White printers. - An A3 may not be sufficient to be a self-explanatory document.

Table A.16: A3 Architecture Overview User Evaluation

Statement 1: It is difficult to create an A3 Architecture Overview							
Strongly Agree	0%	Agree	25%	Disagree	50%	Strongly Disagree	25%
Statement 2: It would have been easier to write a text document...							
Strongly Agree	0%	Agree	0%	Disagree	100%	Strongly Disagree	0%
Statement 3: ... and the value would be similar							
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	25%
Statement 4: I will recommend the approach to other people							
Strongly Agree	75%	Agree	25%	Disagree	0%	Strongly Disagree	0%
Statement 5: Some steps were hard (e.g. creating a functional view)							
Strongly Agree	0%	Agree	75%	Disagree	25%	Strongly Disagree	0%
Statement 6: People find my A3 Architecture Overview more interesting than a document							
Strongly Agree	25%	Agree	75%	Disagree	0%	Strongly Disagree	0%
Statement 7: Having a predefined structure is of help during the creation process							
Strongly Agree	25%	Agree	75%	Disagree	0%	Strongly Disagree	0%
Statement 8: I think there is enough information in my A3 Architecture Overview							
Strongly Agree	50%	Agree	50%	Disagree	0%	Strongly Disagree	0%
Statement 9: I missed the support of a SW tool during the creation process							
Strongly Agree	0%	Agree	0%	Disagree	0%	Strongly Disagree	100%
Statement 10: I think it will help me to communicate better to other stakeholders							
Strongly Agree	25%	Agree	50%	Disagree	25%	Strongly Disagree	0%
Statement 11: I think it is hard to transfer / teach the approach to other people							
Strongly Agree	25%	Agree	0%	Disagree	25%	Strongly Disagree	50%
Statement 12: The creation effort it too high for the benefits it brings							
Strongly Agree	0%	Agree	0%	Disagree	25%	Strongly Disagree	75%
Statement 13: People have problems understanding my A3 Architecture Overview							
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	25%
Statement 14: It is hard to incorporate it in my regular activities							
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	25%
Statement 15: I will keep using the approach in future projects							
Strongly Agree	25%	Agree	75%	Disagree	0%	Strongly Disagree	0%
Statement 16: I see the value of using an A3 sheet instead of several A4 sheets							
Strongly Agree	0%	Agree	100%	Disagree	0%	Strongly Disagree	0%

Strong Points for Creators
- Attractive, visual, limited amount of information but of high quality. - Overview in uniform form, other subsystems look similar (layout). - Overview triggers *"oh yeah, we must not forget ..."* remarks. - Approach is nice, could be used for more than architecture only. - If the output of simulations can be (automatically) incorporated to the A3, the A3s could become interactive, and then I expect will be used very heavily. - Maybe we should have a similar approach for designs as well. - Limited information allowed, guidance in structure and quantification. - Forces one to be terse. - Is graphical in nature.
Weak Points for Creators
- The quality of the A3 depends on the person who creates it. - Tailored to architecture, thus many details cannot be included. - Somehow the A3s give people the impression I know/have all the details on the topic. - Maybe make the A3s digital (hyper linked). - I often encounter is questions like: *"what if we change ..."*. Basically those questions all ask for simulations. Most people are designers and work on details. So very quickly they also ask me for details. - No guarantee: people can make lousy A3s. - Based on paper but the community is electronic. - Cross referencing is a problem.

Table A.17: A3 Architecture Overview Creation Evaluation

Appendix B

Requirements Analysis

In this Appendix we evaluate whether the A3 Architecture Overview supports effective communication in product evolution, and whether it is applicable in an industrial environment. Observations and feedback from the surveys (see Appendix A), presented in Figure 11.1, are used to evaluate whether the A3 Architecture Overview meets the requirements presented in Table 6.2.

B.1 A3 Architecture Overview as an Effective Communication Tool

B.1.1 Practical Industrial Requirements of Communication Tools

Has small overhead — According to the creators of A3 Architecture Overviews, the creation of A3 Architecture Overviews is not difficult. All practitioners who created an A3 Architecture Overview (see Table 10.2) stated that creating an A3 Architecture Overview was easier than creating a text document, and that the task of creating A3 Architecture Overviews can be easily incorporated in their daily activities. In addition, the creation effort was considered not high it for the benefits that having A3 Architecture Overview provides.

The learning process to use and to create A3 Architecture Overviews does not require much time. Almost no guidance is required to create an A3 Architecture Overview[1]. People from different disciplines started creating them with little or no training.

Does not depend on custom-made software tools — The creation, visualization, and use of A3 Architecture Overviews do not require the use of custom-made software tools. The way practitioners use, visualize, or create their A3 Architecture Overview is up to them. Ms Visio, Ms Word, modeling tools, or just paper and pencil are needed to create an A3 Architecture Overview. During the creation of A3 Architecture Overviews, none of the practitioners who created A3 Architecture Overviews missed the support of a software tool.

Provides trusted output — The A3 Architecture Overview guidelines encourages the creators to provide credibility estimations for their input, e.g. such as by using color-coding to differentiate levels of confidence (see Figure 8.5(a)). An A3 Architecture Overview also provides information about the A3 Architecture Overview author such as contact details, as well as information regarding other experts that can be consulted to back up the information provided by the A3 Architecture Overview. The trust on the data however, as indicated by practitioners, depend on the person that has created the A3 Architecture Overview, which can result in poor estimations.

Works even with incomplete input — An A3 Architecture Overview does not need to be complete to be used. Even with incomplete information an A3 Architecture Overview can be used to communicate knowledge; in fact, it is encouraged to use it from the very beginning to

[1]The A3 Architecture Overview Cookbook (see Appendix D) was provided when no training was given

gather additional information. Most A3 Architecture Overview creators stated that their A3 Architecture Overview had enough information to be used (even when the A3 Architecture Overviews were not finished), and that the users of their A3 Architecture Overviews did not have trouble understanding them.

It is easy to use — As stated in Section 10.2, an A3 Architecture Overview is more usable than a traditional document (although this does not automatically mean it is easy to use). In theory, an A3 Architecture Overview is easy to use as it only requires to use a single sheet of paper, which can be printed or visualized on a screen. The majority of A3 Architecture Overview users did not perceive the sheet size as a problem for its use.

During the participation in the different projects at Philips Healthcare MRI (see Section 10.1), A3 Architecture Overviews were sent to experts to be used during meetings and or discussions, with little or no explanation about the A3 Architecture Overview itself. Experts attended the meetings with annotations in their A3 Architecture Overviews that triggered discussions and increased the A3 Architecture Overview content. No explanations about the use of A3 Architecture Overviews was needed during these meetings.

It is appealing — All users stated that they like the idea of having different views within the same A3. Most of them also stated that to learn a new topic, an A3 Architecture Overview is preferred to other means such as text documents, and that reading an A3 Architecture Overviews is preferred to reading a document. Finally, most of them stated that they would like to have an A3 Architecture Overviews of their current work, and of other system aspects. This shows an interest from practitioners in the tool. Those practitioners who created A3 Architecture Overviews stated that they will recommend the approach to other people, and that they will keep using it in their future projects.

B.1.2 Desired Properties of Communication Tools

Provides limited amount of information — The A3 Architecture Overview physical design uses an A3 sheet size. Forcing the A3 size limits the amount of information that can be provided. From the feedback obtained from the A3 Architecture Overview users, we see that they appreciate using an A3 instead of several sheets to display knowledge. We also see that the A3 size displays an adequate amount of information for most users. Creators of A3 Architecture Overviews (see Table 10.2) stated the value of using an A3 when consolidating their knowledge.

Uses visual representations — The A3 Architecture Overview design encourages the use of visual representations to display architecture information. In addition, the A3 Architecture Overview provides explicit visual aids to support the understanding of the information within the A3 Architecture Overview. According to users, the visual aid(s) was the A3 Architecture Overview element was most useful. Creators of A3 Architecture Overviews also stated that a strong point of the tool is that *"it is attractive, visual, and limited amount of information but of high quality"*, that *"it is graphical in nature"*, etc.

Uses an appropriate size to display complex information — An A3 fits well within the field of view, even if this field is reduced in the presence of complex information. Experiences in the use of other sizes during real projects showed that the A3 is better size to display complex information that bigger or smaller size (this is also supported by Toyota's A3 Reports, see Section 5.2.2).

Keeps the notation simple — The A3 Architecture Overview does not use standard notations but a simple one to represent architecture knowledge, such as natural language and visual representations. Most users of A3 Architecture Overviews stated that the notation used in the A3 Architecture Overviews provided was not difficult to understand. However, half of architects, which are also users of the A3 Architecture Overview, did not find the notation easy.

When compared with traditional documents, an A3 Architecture Overview is easier to understand. Creators of A3 Architecture Overviews stated that people who used their A3 Architecture Overviews did not have troubles understanding them.

Limits the amount of visual attributes and ensures differences among them — The guidelines to create readable A3 Architecture Overviews provided in Section 9.5 limit the amount of visual attributes such as shapes, colors, etc, to a comfortable number. In addition, those guidelines state the need for explicit differentiation among attributes, such as using different shapes for functional and physical elements.

A simple way to ensure differentiation among visual attributes is by using color, which is one of the ways in which A3 Architecture Overviews difference attributes. Users of A3 Architecture Overviews stated that the use of color was useful when reading A3 Architecture Overviews. However, as stated by some users, color coding should be used with care, as color clutter may happen if the colors are not chosen wisely.

Keeps a consistent way of communication — The A3 Architecture Overview uses a consistent way of communication. First by having a fixed physical design (an A3), and second by providing a predefined format to display the architecture knowledge. Users stated that the A3 size is not a problem for its use, and that an A3 is more valuable than using several A4. They also stated that having a predefined format in the A3 was of great help for the reader when finding the information, making it easier to read other A3s. Creators of A3 Architecture Overviews also stated that they saw the value in using an A3 to capture their knowledge, as well as having a predefined structure during the creation process.

Provides a shared view — Instead of creating different representations for different stakeholders, the A3 Architecture Overview aims to provide a single shared multi-view of the system that is understood by a wide variety of stakeholders. For understanding, as stated in previous paragraphs, the notation is kept simple and the use of visual attributes encouraged. Users of A3 Architecture Overviews stated that they would use an A3 Architecture Overview as a collaboration tool with other disciplines, as it is easier to understand than traditional documents, and the visual attributes help in shared understanding.

Enables a flexible way to share knowledge — A flexible knowledge sharing tool should be easily adapted to cope with the user's preferred ways to share knowledge. As shown in Chapter 10, the A3 Architecture Overview was adapted to cope with different ways to share knowledge. The A3 Architecture Overview enables creators to use their preferred style, visualization methods (e.g. 3D model for the physical view), etc, to share architecture knowledge.

Improves existing written mechanisms — The A3 Architecture Overview is a written mechanism. If we consider existing written mechanism traditional documents (and not i.e. Wikis), we showed in Section 10.2 that A3 Architecture Overviews when compared with traditional documents they were perceived to be more readable, usable, understandable and

with a more adequate amount of information. This means that in this context, we can consider A3 Architecture Overviews as an improvement over existing written mechanism.

B.2 A3 Architecture Overviews Tailored to the Architecting Process

Supports the creativity of architects — As shown in the A3 Architecture Overviews examples of Section 10.4 (see Figures 10.4, 10.5, 10.9, 10.11), each practitioner applied a different style when creating his A3 Architecture Overview. It can be observed that although all them followed the guidelines to create A3 Architecture Overviews, they could still apply their preferred styles. We can observe for example the use of black&white, different fonts, preferred modeling tools (e.g. 3D CAD models), etc, resulting in personalized A3 Architecture Overviews. Although we cannot state that the A3 Architecture Overview directly supports the creativity of architects, the A3 Architecture Overview does not constrain it by allowing enough flexibility to represent knowledge in a flexible way.

It is used by a wide variety of stakeholders — During the projects performed at Philips Healthcare the A3 Architecture Overview was used to share knowledge with a wide variety of stakeholders. Different stakeholders were able to use it without problems, despite their background or experience. It was stated by most users from different disciplines that they would use the A3 Architecture Overview as a collaboration and communication tool with other stakeholders.

Does not take too much time from architects — Creating an A3 Architecture Overview is easier than creating a text document according to all practitioners who created an A3 Architecture Overview. All of them stated that having a predefined structure was of great help during the creation process. They all stated that it is not hard to incorporate the creation of A3 Architecture Overviews into their regular activities, and consequently they all will keep using the approach in the future. The time required to create an A3 Architecture Overview diminishes with practice. It may take only a few hours to create an A3 Architecture Overview for an experienced practitioner.

Encourages the dissemination of knowledge — As stated in Section 5.1.2, one of the main reasons why knowledge is not shared -and consequently disseminated- is because it is considered a hard process. The A3 Architecture Overview aims to provide a simple (yet effective) way to consolidate knowledge. According to practitioners, creating them is not hard, and the value they obtain from the A3 Architecture Overview is high for the effort required to create them. They all stated that they will keep using the tool in future projects, and that they will recommend the tool to other people. Philips Healthcare MRI management encourages now the creation of A3 Architecture Overviews to project leaders.

On the other side, the A3 Architecture Overview aims to make the knowledge easy to read and appealing, so the stakeholders receive the knowledge. As stated before (see Section B.1.1), the A3 Architecture Overview is appealing to users, therefore, encouraging them to read the knowledge provided. During the projects at Philips Healthcare MRI, people attended the meetings having read the A3 Architecture Overviews provided. Most users stated that they would like to have more A3 Architecture Overviews; both from their current work, and from other system aspects.

B.3 A3 Architecture Overviews Support the Needs of Architects

Delivers the right information to the stakeholders while keeping the irrelevant part low — An A3 Architecture Overview limits the amount of information that can be delivered, encouraging to keep only the essential information. Focus is on delivering the right information, while irrelevant information is excluded due to the limits imposed by the A3. Users stated that among the strong points of the A3 Architecture Overview there is the fact that it provides limited information but of high quality, and that forces to be specific.

Ensures that the information is conveyed and interpreted correctly — The A3 Architecture Overview uses natural language and visual representations to avoid the need for domain or specific modeling languages. By using visual representations, the A3 Architecture Overview aims to remove the ambiguity that purely textual representations may have.

By using simple notation, the A3 Architecture Overview aims to make the information contained easier to read (see Section B.1.2). Although most A3 Architecture Overview users stated that the notation was not complicated, half of the architects stated that the notation was sometimes complicated (no other stakeholder stated is was complicated). As the A3 Architecture Overview is targeted mainly to architects, the A3 Architecture Overview may have not fully met this requirement[2].

Records changes to the architecture knowledge repository — As Stated in Section 8.3, a set of A3 Architecture Overviews forms a repository of architectural knowledge. By knowing which A3 Architecture Overviews are changed, changes to the architecture knowledge repository can be recorded. There are however some issues that that makes difficult to meet this requirements. As stated by some practitioners, cross referencing among A3 Architecture Overviews may lead to problems when changing the repository; a change in one A3 Architecture Overview may require to update other A3 Architecture Overviews (e.g. the reference section), leading to some problems when changing the architecture knowledge repository. As this is a manual process, required changes may be overlooked if the responsible of the repository is not aware of them.

Retrieves architectural knowledge stored in the head of people — The Reverse Architecting process along with the A3 Architecture Overview aims to make implicit knowledge explicit. By being easy to create, of high value, people can incorporate the task of creating A3 Architecture Overviews into their regular activities. In addition, as shown in Figure 10.6, an A3 Architecture Overview enables to capture the knowledge produced at meetings or discussions, enabling to capture what is stored in the head of people.

Enables reusing knowledge from previous experiences and products in current developments — To support the evolution of complex systems is the ultimate goal of an A3 Architecture Overview. For that, focus of the A3 Architecture Overview is on architecture knowledge (see Section 4.2.1). By making this knowledge explicit, it can be reused in new generations of products.

Helps keeping a structured overview of what has been communicated with a stakeholder — If A3 Architecture Overviews are used as means of communication with the stakeholders, it is easy to know what has been communicated to the stakeholders -the contents of the A3 Architecture Overviews used-. By knowing which A3 Architecture Overviews have been used

[2]This clearly supports the statement of Section 4.1.2, in which we found that architects look for -very- simple solutions.

to communicate with the stakeholders, it is possible to keep an overview of what has been communicated with them.

B.4 A3 Architecture Overviews Mitigate Evolution Barriers

Supports management of system complexity — A3 Architecture Overviews provide manageable sets of information. The limited dose of information each A3 Architecture Overview provides aims to prevent information overload. By partitioning the system knowledge in small doses, the A3 Architecture Overview tries to help managing system complexity by making it easier to digest the large amount of information to the user.

Prevents lack of system overview — An A3 Architecture Overview is designed to provide only the essential information, it is to say, an overview of the needed information. In addition to that, one of its goals is to maintain a system view (see Section 8.1), for that, an A3 Architecture Overview provides several views within the same sheet of paper, which is also a way to provide overview (all the views can be visualized at a glance). Most users consider that an A3 Architecture Overview has an adequate amount of information and all of them liked the idea of combining several views within the same A3.

Deals with ineffective knowledge sharing — As stated in Section 4.2.2, most approaches to support knowledge sharing fail mainly because they do not deliver all relevant architecture knowledge such as design decisions and rationale, and because they are developed from a technology perspective -which is usually not the preferred way of architects- without ensuring that the knowledge is effectively communicated to the stakeholders.

An A3 Architecture Overview delivers all the types of information that belongs to the architecture knowledge (see Section 4.2.1), and it is structured in a fashion that ensures that all elements of architecture knowledge have a place within the A3. In addition, an A3 Architecture Overview is a written approach -therefore not developed from a technology perspective- that takes into account human and organizational factors that affect communication and provide means to cope with them.

Helps finding the required system information — Information is hard to find because it has no meaningful structure [Koniger and Janowitz, 1995]. An A3 Architecture Overview aims to gather all relevant system information in one physical place. In addition, it provides a clear structure to allocate different types of information within the A3. Most of the users valued having the system information within the same A3 instead of having it spread within several A4, and stated that having a predefined structure was of help while reading A3 Architecture Overviews.

Supports communication across disciplines and departments — As stated in Section 5.1.2, one problem of communication is the perturbation produced by "architecture noise". In addition to the architecture noise, another problem of communication is the lack of a shared model of the system. As stated before, an A3 Architecture Overview is designed to provide a shared model, in a fashion that avoids -to some extent- architecture noise. As an A3 Architecture Overview is meant for easy use, it can be taken to meetings and used as a tool to communicate. Most practitioners who created an A3 Architecture Overview for their work believe that an A3 Architecture Overview will help them to communicate better with other disciplines and departments.

B.5 A3 Architecture Overviews Deal with Observed Evolution Challenges

Supports moving from incremental development to top-down architecting — As stated in Section 3.4.2, every system has an architecture. An architecture, once consolidated in an architecture representation is a key artifact to support the architecting process. The A3 Architecture Overview focuses on architectures, and is tailored to the architecting process. An A3 Architecture Overview is meant to assist architects in their duties, and to cover their specific needs. Once A3 Architecture Overviews are created, they can be used as an architecting tool that architects can use to architect new generation of systems. Having the architecture explicit enables the process of moving from bottom-up development to top-down architecting.

Reduces learning curve — Reading documents outside one's domain of expertise is a challenge, and the time people can spend reading documents to learn about new topics is very limited. An A3 Architecture Overview has been pointed by users as better way for learning about a new topic than other means such as documents. It was stated by users that they prefer to read an A3 Architecture Overview than a document, especially when learning about a new topic. In addition, a visual representation is usually more effective for learning than plain text, as it is retained better by the reader [Koning, 2008]. As an A3 Architecture Overview encourages the use of visual representations, it is probably best suited for learning than purely text documents. According to the users, the visual aids within the A3 Architecture Overview are the most valuable element.

Supports to estimate the impact of change — Estimate the impact of change is hard. It is also difficult to visualize the impact of a change in the system for a large part of the practitioners. An A3 Architecture Overview provides different views of a specific system aspect, from a system point of view. An A3 Architecture Overview enables to foresee the impact that a change may have on the system from different views (e.g. functionality, physical), and provides numbers to estimate the impact to the key parameters. As an A3 Architecture Overview is mostly visual, it also helps visualizing the impact. Most A3 Architecture Overview users stated that the A3 Architecture Overview can be used to estimate the impact of change.

Deals with the mono-disciplinary focus of developers — An A3 Architecture Overview brings together the concerns of many stakeholders, with a notation that can be understood by many disciplines. In addition, as an A3 Architecture Overview provides a system view, forcing developers to look into other domains to get the complete picture of the issue at hand, rather than a localized view.

Supports repartitioning the system — An A3 Architecture Overview makes architectures explicit. By having an explicit artifact representing the architecture, the architect is able to make better and informed decisions when repartitioning the system. The architect can also communicate the repartitioning better, as well as to estimate the impact of that the repartitioning may have on the system.

Appendix C

Frequently Asked Questions (FAQ)

Adopting new approaches and tools may produce some confusion at the beginning. The A3 Architecture Overview is no exception. In this appendix we provide some Frequently Asked Questions (FAQ) rose when the A3 Architecture Overview has been introduced in a new project or company. With this we hope to provide some insight on what reaction to expect from people when introducing A3 Architecture Overviews.

- *Does the A3 Architecture Overview tool aims to replace existing documentation?* No. It may happen that as a consequence of creating A3 Architecture Overviews some documents are no longer needed, yet the A3 Architecture Overview tool aims to support communication by capturing architectural knowledge. It provides an overview, not plenty of detailed information. Documents are needed, and the A3 Architecture Overview tool helps by giving just "small" doses of what is present in some of those documents and implicit in experts' minds. Ideally, an A3 Architecture Overview will be 'on the top' of the pile of documents of a specific system aspect, to be used as a starting point for learning, and as a way to communicate that system aspect with different stakeholders.

- *How to fit all architectural knowledge of a system aspect in an single A3 Architecture Overview?* You cannot. The aim is not to fit **all** knowledge but the essential one. The A3 size and its structure forces the author to choose among all bits and pieces of information in order to keep only the essential one. It may happen however that a few A3 Architecture Overviews are needed to describe different viewpoints of the system aspect.

- *Who should create A3 Architecture Overviews?* In theory anyone who needs to consolidate and communicate specific knowledge. In case of the architectural knowledge architects are expected to create most of them. However, as pointed out in Section 4.1.2, we found that architects are great consumers of architectural knowledge but are reluctant to consolidate theirs. However, as consumers of architectural knowledge they are usually willing to review, modify and correct those A3 Architecture Overviews created. To support architects to consolidate their knowledge a new role may be required (e.g. A3 Architecture Overview creator).

- *When do I know that my A3 Architecture Overview is complete?* It will probably be never complete. The more the A3 Architecture Overview is used and the more it is reviewed the more people will find things that should be changed, added, modified, etc. However it should be good enough to be used as a communication tool after a the first iteration (see Section 9.3). If after a few iterations the contents of the A3 Architecture Overview barely changes and only minor modifications are required, at that point the A3 Architecture Overview is probably complete enough (for the time being).

- *Do I need to follow the step-wise guide and the proposed reverse architecting process?* Not necessarily. The guide and the reverse architecting process are aimed to provide a

systematic way of creating A3 Architecture Overviews, however when practitioners become proficient with the A3 Architecture Overview creation process, it becomes easier to start directly modeling in an A3 template instead of by using a step-by-step process.

- *Why not using a modeling language such as UML in the A3 Model?* From the experiences using modeling languages (see Chapter 6), we have experienced that when communication with different disciplines and departments, modeling languages are more a barrier than a support. From the survey at Philips (see Appendix A), we found that the use of modeling languages is only desirable by employees with background in software (e.g. software engineers, software architects). As the development of complex systems such as the MRI require multidisciplinary teams with different backgrounds, it is unlikely that all members are familiar with a specific modeling language.

Appendix D

A3 Architecture Overview Cookbook

To support the creation of A3 Architecture Overviews, an A3 Architecture Overview Cookbook was developed (which is an A3 Architecture Overview itself, as shown in Figure D.1). The goal was twofold; first provide practitioners with an step-wise guide to support the creation of A3 Architecture Overviews (as described in Chapter 9), and secondly to provide an A3 Architecture Overview example. Embedded in the cookbook (in the digital version) was MS Visio templates created to support the process.

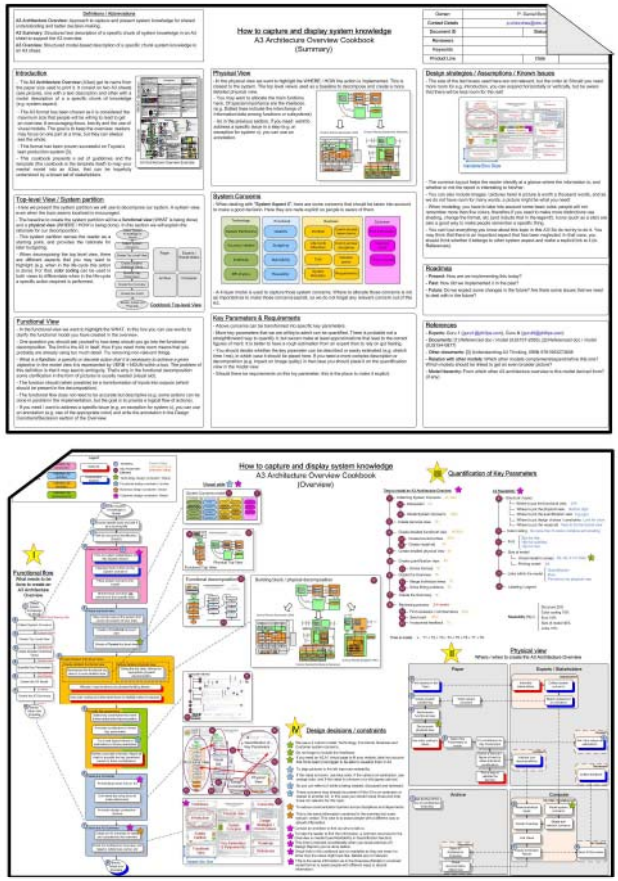


Figure D.1: A3 Architecture Overview Cookbook

Acknowledgments

I may have written this Thesis, however that does not mean that I deserve all the credit. The truth is that without the collaboration of many people, this Thesis would have never been possible. I want to thank all those persons, and many more that may not be present in this section, for their support over the last 4 years.

First of all I want to thank my supervisor, **Maarten Bonnema**, not only for giving me the opportunity to do the Ph.D., but for all the long hours he has spent with me, giving me direction, support, and motivation to continue. He has been a friend over the last four years, and probably without him I would have gave up when this research seemed impossible. He is the kind of supervisor all Ph.D.s should have.

I also want to thank **Phil van Liere** for all the time he has spent mentoring me, teaching me, and reviewing my material. He has been my best ally in the complex MRI world. I have been lucky to have him as my mentor at Philips. Thanks to his incredible knowledge and sharp vision I was able to understand the complexities of the MRI and to understand what is useful for a system architect. Thanks to him I have had a great time at Philips Healthcare.

Probably this Thesis would have been completely different without the inspiration provided by **Gerrit Muller**. Gerrit challenges every aspect of your work, and forces you to see things in a totally different perspective -which usually is a better one-. Working with Gerrit has make grow as a professional in the Systems Engineering field, and I am thankful to him for that. It has been a great experience to work with him.

Without the daily support of many other people, I probably had not endured to see the end of the long Ph.D. journey. For that, I want to thank **Trosky Callo** for being a great colleague and friend, it has been great to work with you; I also want to thank **Seray Candar**, for her company and for making writing the Thesis more enjoyable; my good friends **Carlos Arias** and **Jesús Fernandez** for always listening to my complaints; the Darwin team, especially **David Watts**, for being a great manager easy to work with, **Alexander Douglas**, for being such a passionate guy about pure research, and **Pierre van de Laar**, for being always willing to give a hand; and finally, to all the amazing people I have met in Eindhoven; **Anna Sues, Cesar Lopez, Gosia Perz, Olga Urbaniak, Lesly Garcia, Oscar Perez, Armando Ortiz**, and many others.

I have to make a special mention to **Inge dos Santos**, the secretary of the CTW department. She has diligently and efficiently solved all the problems that raised during the past years. Thanks to her I haven been able to live at Eindhoven while still be part of Twente University.

Last but not least, I want to thank my family. They are really the architects of my past and present achievements. They have always motivated me, guided me, and cared about me no matter what. My father, **Felipe Daniel**, has always encouraged me to give the best of me and to keep going no matter the circumstances. My mother, **María Antonia**, has showed to me that you achieve anything you desire, as long as you work hard. And finally my bother **Pablo**, who has showed me that you have to put your heart in what you are doing. I hope I have made them proud.

I will always cherish this last four years as one of the best periods of my live. I have had so many experiences, and I have really enjoyed them all. Although I feel sad for the period that ends, I am also excited about the new challenges to come. I hope to see you all there.

About the Author

P. Daniel Borches was born in Madrid (Spain). He got his Masters degree in Telecommunication Engineering at the Polytechnic University Carlos III of Madrid in 2004.

P. Daniel Borches has always been interested in the technology sector, specifically the mobile and medical sector. During his master Thesis he combined experiences in both sectors by designing a system to transmit the EKG of patients with a heart disease to the doctor located at the hospital without the need to admit the patient at the hospital for continuous monitoring by using state-of-the-art mobile and web technologies, which lead to a grant to work at Nokia. This master Thesis was honored with distinction. After his experience at Nokia he worked as a mobile consultant at one of the largest Spanish mobile network operators; Telefónica.

In 2006 he moved to The Netherlands and joined the University of Twente to get his Ph.D. at Philips Healthcare. The research field was systems architecting and systems engineering applied to the industrial sector. During his research P. Daniel Borches has been active in the Systems Engineering community by publishing and giving lectures at INCOSE and CSER symposiums. The outcome of his research work has attracted the attention of large companies such as ASML, Daimler, FEI, and Boeing which are currently applying the A3 Architecture Overview tool into their development process.

Publications during the research

Borches, P. D. and G. M. Bonnema: 2008; *"Living" Architecture Overviews - Supporting the Design of Evolutionary Complex Systems*; in CIRP Design Seminar; Twente, The Netherlands.

Borches, P. D. and G. M. Bonnema: 2008; *On the Origin of Evolvable Systems: Evolvability or Extinction*; Proceedings of the TMCE; vol. 2: pp. 1351-1353.

Bonnema, G. M. and P. D. Borches: 2008; *Design with Overview - How to Survive in Complex Organizations*; in Proceedings of the 18th Annual Symposium of the International Council of Systems Engineering (INCOSE'08); Utrecht, The Netherlands.

Borches, P. D. and G. M. Bonnema: 2009; *Coping with System Evolution - Experiences in Reverse Architecting as a Means to Ease the Evolution of Complex Systems*; in Proceedings of the 19th Annual Symposium of the International Council of Systems Engineering (INCOSE'09).

Borches, P. D. and G. M. Bonnema: 2010; *A3 Architecture Overviews, Focusing Architectural Knowledge to Support Evolution of Complex Systems*; in Proceedings of the 20th Annual Symposium of the International Council of Systems Engineering (INCOSE'10).

Borches, P. D. and G. M. Bonnema: 2010; *System Evolution Barriers and How to Overcome Them!*; in Proceedings of the 8th Annual Conference on Systems Engineering Research; pp. 455-464.

Definitions / Abbreviations

A3 Architecture Overview: Approach to capture and present system knowledge for shared understanding and better decision-making.

A3 Summary: Structured text description of a specific chunk of system knowledge in an A3 sheet to support the A3 overview.

A3 Overview: Structured model-based description of a specific chunk system knowledge in an A3 sheet.

How to capture and display system knowledge

A3 Architecture Overview Cookbook (Summary)

Owner:	P. Daniel Borches		
Contact Details	p.d.borches@ctw.utwente.nl		
Document ID		Status	DRAFT (v1)
Reviewers			
Keywords			
Product Line		Date	

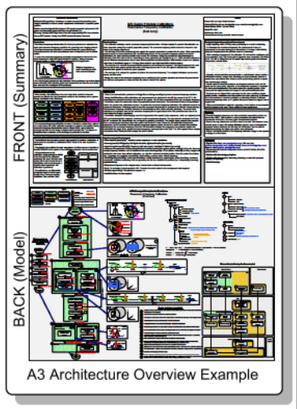
Introduction

- The **A3 Architecture Overview** (A3ao) get its name from the paper size used to print it. It consist on two A3 sheets (see picture), one with a text description and other with a model description of a a specific chunk of knowledge (e.g. system aspect).

- The A3 format has been chosen as it is considered the maximum size that people will be willing to read to get an overview. It encouraging focus, brevity and the use of visual models. The goal is to keep the overview; readers may focus on one part at a time, but they can always see the whole.

- This format has been proven successful on Toyota's lean production system [3].

- This cookbook presents a set of guidelines and the template (the cookbook is the template itself) to map your mental model into an A3ao, that can be hopefully understood by a broad set of stakeholders.



A3 Architecture Overview Example

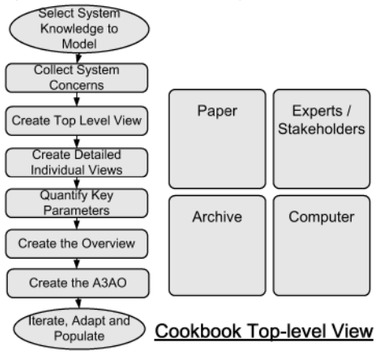
Top-level View / System partition

- Here we present the system partition we will use to decompose our system. A system view, even when the topic seems localized is encouraged.

- The baseline to create the system partition will be a **functional view** (WHAT is being done) and a **physical view** (WHERE / HOW is being done). In this section we will explain the rationale for our decomposition.

- This system partition serves the reader as a starting point, and provides the rationale for later budgeting.

- When decomposing the top level view, there are different aspects that you may want to highlight (e.g. when in the life-cycle this action is done). For that, **color coding** can be used in both views to differentiate when in the life-cycle a specific action required is performed.



Cookbook Top-level View

Functional View

- In the functional view we want to highlight the WHAT. In this box you can use words to clarify the functional model you have created in the overview.

- One question you should ask yourself is how deep should you go into the functional decomposition. The limit is the A3 in itself, thus if you need more room means that you probably are already using too much detail. Try removing non-relevant things.

- What is a **function**: *a specific or discrete action that it is necessary to achieve a given objective.* In the model view it is represented by VERB + NOUN within a box. The problem of this definition is that it may lead to ambiguity. That's why in the functional decomposition some clarification in the form of pictures is usually needed (visual aid).

- The function should (when possible) be a transformation of inputs into outputs (which should be present in the decomposition).

- The functional flow does not need to be accurate but descriptive (e.g. some actions can be done in parallel in the implementation, but the goal is to provide a logical flow of actions).

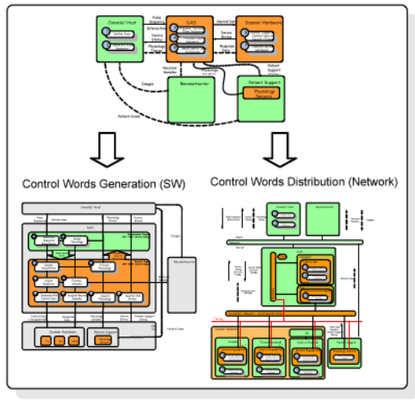
- If you need / want to address a specific issue (e.g. an exception for system x), you can use an annotation (e.g. star of the appropriate color) and write the annotation in the Design Constraint/Decission section of the Overview.

Physical View

- In the physical view we want to highlight the WHERE / HOW the action is implemented. This is closest to the system. The top level view is used as a baseline to decompose and create a more detailed physical view.

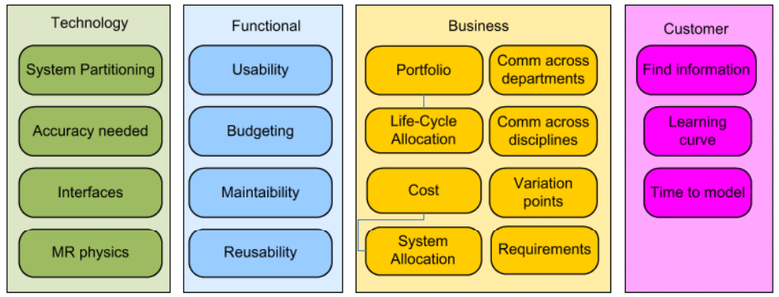
- You may want to allocate the main functions here. Of special importance are the interfaces. (e.g. Dotted lines indicate the interchange of information/data among functions or subsystems)

- As in the previous section, if you need / want to address a specific issue in a step (e.g. an exception for system x), you can use an annotation.



System Concerns

- When dealing with **"System Aspect X"**, there are some concerns that should be taken into account to make a good decision. Here they are made explicit so people is aware of them.



- A 4 layer model is used to capture those system concerns. Where to allocate those concerns is not as important as to make those concerns explicit, so we do not forget any relevant concern out of this A3.

Key Parameters & Requirements

- Above concerns can be transformed into specific key parameters.

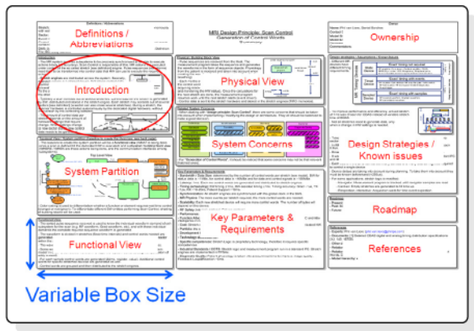
- More key parameters that we are willing to admit can be quantified. There is probable not a straightforward way to quantify it, but we can make at least approximations that lead to the correct figures of merit. It is better to have a rough estimation from an expert than to rely on gut feeling.

- You should decide whether the key parameter can be described or easily estimated (e.g. stretch time 1ms), in which case it should be placed here. If you need a more complex description or decomposition (e.g. impact on image quality) in that case you should place it on the quantification view in the model view.

- Should there be requirements on this key parameter, this is the place to make it explicit.

Design strategies / Assumptions / Known Issues

- The size of the text boxes used here are not relevant, but the order is! Should you need more room for e.g. introduction, you can expand horizontally or vertically, but be aware that there will be less room for the rest!



- The common layout helps the reader identify at a glance where the information is, and whether or not the report is interesting to him/her.

- You can also include images / pictures here! A picture is worth a thousand words, and as we do not have room for many words, a picture might be what you need!

- When modeling, you have to take into account some basic rules: people will not remember more than five colors, therefore if you need to make more distinctions use shading, change the format, etc (and include that in the legend!). Icons (such as a star) are also a good way to make people remember a specific thing.

- You can't put everything you know about this topic in this A3! So do not try to do it. You may think that there is an important aspect that has been neglected. In that case, you should think whether it belongs to other system aspect and make a explicit link to it (in References).

Roadmap

- **Present:** How are we implementing this today?

- **Past:** How did we implemented it in the past?

- **Future:** Do we expect some changes in the future? Are there some issues that we need to deal with in the future?

References

- **Experts:** Guru 1 (guru1@philips.com), Guru N (guruN@philips.com)

- **Documents:** [1] Referenced doc / model (XJS157-2555), [2] Referenced doc / model (XJS154-0877)

- **Other documents:** [3] Understanding A3 Thinking, ISBN 9781563273605

- **Relation with other models:** Which other models complement/expand/refine this one? Which models should be linked to get an even broader picture?

- **Model hierarchy:** From which other A3 architecture overview is this model derived from? (if any)

How to capture and display system knowledge

A3 Architecture Overview Cookbook (Overview)

Legend

Function
Data / information exchange
Validation by consensus
Validation by architect
Validation by expert(s)
Validation by measurements
Optional
Cooperation needed
1 Workflow
1 Key Parameter Element
Known Value
Estimated Value
Unknown Value
Technology design constraint / choice
Functional design constraint / choice
Business design constraint / choice
Customer design constraint / choice

I Functional flow

What needs to be done to create an A3 Architecture Overview

1. Select System Knowledge to Model
   - A3AO Title & Running Title
2. Collect System Concerns
   - 4 column model
3. Create Top Level View
   - Top level view
4. Create Detailed Individual Views
   - Detailed views
5. Quantify Key Parameters
   - Key parameter quantification
6. Create the A3 Model
   - A3 Model
7. Create the A3 Summary
   - A3AO
8. Iterate, Adapt and Populate

1 Select System Knowledge to Model
1 Choose specific topic and add it as a running title
1 Ask for document identification (DocID)

2 Collect System Concers
- Find and select stakeholders of this System Aspect
- Interview them to find out the System Concerns
- Place system concerns into model
- Shade those concerns not relevant for this specific topic

3 Create top level view
- Take a broad view of the system that covers all aspects of your topic
- Create a Functional top level view
- Create a Physical top level view

4 Create Detailed Individual Views
- Create detailed functional view: Decompose the functional top view in a more detailed view
- Create detailed physical view: Using the top view, choose an appropriate physical decomposition
- Allocate / map functions into physical building blocks
- Use color coding and other techniques to highlight relevant aspects

5 Quantify key parameters
- Select key parameters that need further elaboration/decomposition
- Annotate contributions to those key parameters
- Try to add typical values or estimations to those parameters
- Derive or provide a formula / figure of merit to quantify the key parameter based on those contributions

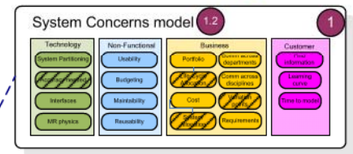
6 Create the A3 Model
- Fit individual views into an A3
- Link views (by using icons to make references)
- Annotate design constraints / choices

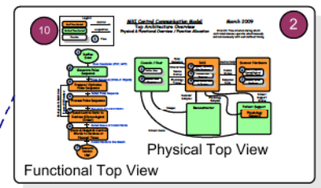
7 Create the A3 Summary
- Create an A3 summary to clarify and complement the overview
- Finish A3 Architecture Overview; add legend, references, owner, etc

8 Iterate, Adapt and Populate

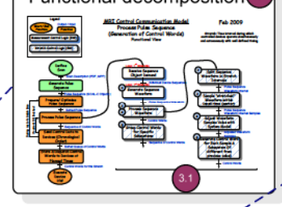
Visual aids

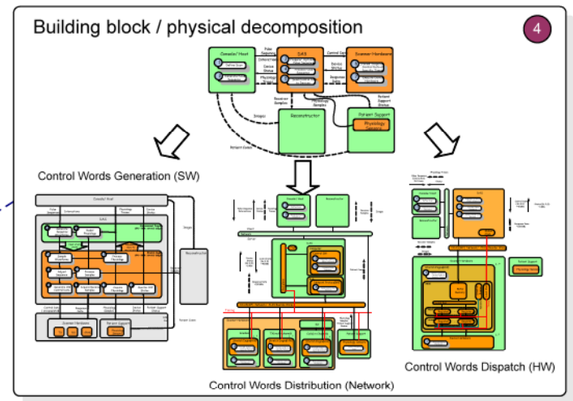
System Concerns model

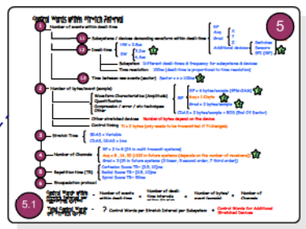


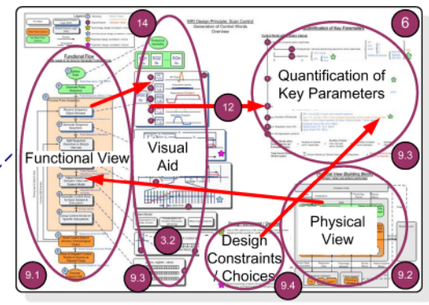


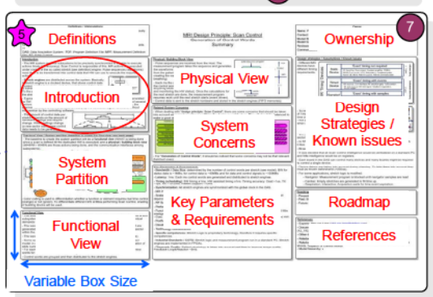
Functional decomposition











III Quantification of Key Parameters

Time to model an A3 Architecture Overview

1. Collecting System Concerns 2h 30m
   - 1.1 Discussion 2h
   - 1.2 Model System Concerns 30m
2. Create top level view 2h
3. Create detailed functional view 4h30m
   - 3.1 Create functional flow 30m
   - 3.2 Create visual aid 2h
4. Create detailed physical view 4h
5. Create quantification view 4h
   - 5.1 Derive formula 1h
6. Create the Overview 3h
   - 6.1 Merge individual views 1h
   - 6.2 Solve fitting problems 2h
7. Create the Summary 3h
8. Reviewing process 2-4 weeks
   - 8.1 Find reviewers / commentators 30m
   - 8.2 Send mail 20m
   - 8.3 Incorporate feedback 1h

Time to model ≈ T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8

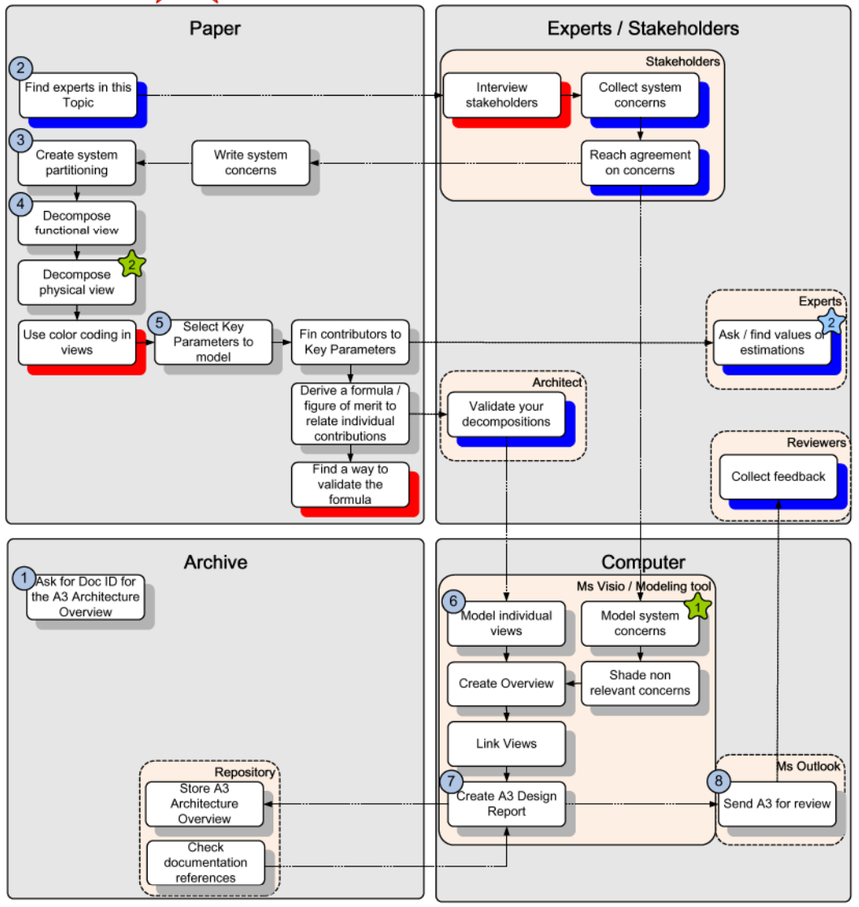
A3 Readability

9. Structure / layout
   - Where to put the functional view: Left
   - Where to put the physical view: Bottom right
   - Where to put the quantification view: Top right
   - Where to put design choices / constraints: Look for room
   - Where to put the visual aid: Next to the functional view
10. Color coding: No more than 5 colors, combine with shading
11. Font: 30pt for title; 18pt for subtitles; 14pt for rest
12. Size of model
    - Virtual model in screen: A3, A2, A1 in Visio
    - Printing model: A3
13. Links within the model: Quantification; Flow; Functions into physical view
14. Labeling / Legend

Readability (%) ≈
- Structure 20%
- Color coding 10%
- Font 10%
- Size of model 40%
- Links 10%

II Physical view

Where / when to create the A3 Architecture Overview



IV Design decisions / constraints

- We use a 4 column model; Technology, Functional, Business and Customer system concerns.
- Do not forget to include the interfaces
- If you need an A2,A1 virtual page to fit your models, take into account that fonts need to be bigger to be able to visualize them in A3
- To align pictures to the left improves redeability
- If the value is known, use blue color, if the value is an estimation, use orange color, and if the value is unknown or a wild guess use red.
- So you can refer to it while is being created, discussed and reviewed.
- These concerns may already be present if this A3 is an extension or related to another A3. In this case you should reuse those and strip those not relevant for this topic.
- To reduce communication barriers across disciplines and deparments.
- This is the same information contained in the overview but more textual / verbal. This view is to assist people with a different way to absorb information.
- Contact an architect to find out who to talk to.
- To help the reader to find the information, a common structure for the Overview is needed (see Readability in Quantification Section)
- This time is reduced considerably when you reuse previous A3 Design Reports you've done before.
- Visual Ads in this cookbook are not readable as they are meant to show how the views might look like. Details are not relevant.
- This is the same information as in the Overview (Model) in a textual/ verbal format to assist people with different ways to absorb information.