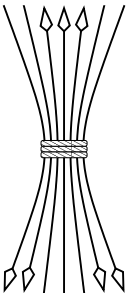
Democratic Processing

Mastering the complexity of communicating systems



Hylke W. van Dijk

Democratic processing

Mastering the complexity of communicating systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 22 november 2004 om 13:00 uur

door

Hijlke Watze VAN DIJK

elektrotechnisch ingenieur
geboren te Wommels.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. H.J. Sips
Prof.dr.ir R.L. Lagendijk

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter	
Prof.dr.ir. H.J. Sips,	Technische Universiteit Delft, promotor
Prof.dr.ir. R.L. Lagendijk,	Technische Universiteit Delft, promotor
Prof.dr.ir. E.F. Deprettere,	Universiteit Leiden
Prof.dr. P.H. Hartel,	Universiteit Twente
Prof.dr.ir. P.H.N. de With,	Technische Universiteit Eindhoven
Prof.dr. W.G. Vree,	Technische Universiteit Delft
Prof.dr.ir. N.H.G. Baken,	Technische Universiteit Delft

BIBLIOGRAFISCHE DATA

Dijk, H.W. van

Democratic processing: Mastering the complexity of communicating systems
H.W. van Dijk. -
Thesis Delft University of Technology – With references. – With summary in Dutch.
November, 2004
ISBN 90 6464 660 0
Subject headings: QoS; Context Aware Processing, Multidisciplinary optimisation
Multiobjective optimisation, Democratic Processing, Communicating systems

Printed in the Netherlands.

Contents

Summary

MODERN communication systems tend to include more and more functionality while offering the user the freedom of mobility. Context-awareness is regarded a great asset for these type of devices. Great care must be taken when developing these devices and their applications, since mobile devices must be small and consequently have limited resources. Fortunately the active function repertoire of a mobile device is smaller than the potential repertoire, hence functionality can be swapped on demand. Given the current state of the technology, a successful development methodology for these kind of systems therefore acknowledges the fact that communication systems are in a constant state of flux. This is even more true when considering the changing conditions of the immediate environment of the system.

Problem

In this dissertation we address the issue of the development and operation of complex communicating systems. The complexity of a system is mainly attributed to irregular relations among the heterogeneous components of a system. An important aspect of a complex communicating system is its being in a constant state of flux. Given this setting of the system we found that flexibility is as much a design parameter as for instance power dissipation.

In this dissertation we focus on a multidisciplinary research programme, the Ubicom (Ubiquitous communications) project [Lagendijk, 2000a]. This research project aimed at developing visual mobile augmented reality, anytime, anywhere, anyplace. The multidisciplinary character of the programme significantly complicates the interaction among individual system components, because each discipline involved brings its own culture, preferred development methodology, language, etc.

Our main goal is to develop a framework to organise the coordination among system components of a communicating system. The Ubicom system is our motivating case study.

Approach

Our approach is based on a three notions. First, a system has many beyond our reach properties, which are valued differently by different stakeholders. We adopt an "ontology of the world" [Bunge, 1977] to put in perspective different views and different predicates of the system.

Second, we conjecture that a complex system of communicating entities should be addressed as an organism with distributed coordination. Rather than applying a hierarchical organisation we suggest a heterarchic organisation of the coordination, where the dominance is demand driven. Components in the system therefore should be able to *abstract* their distortion, capacity, and resource utilisation metrics, should be able to *adapt* to changes in their milieu, and should be willing to *cooperate* with components in their immediate environment; all components should, like the system, be truly context aware. The above arguments are merged into a framework (ARC) for coordinating a communicating system.

Third, the compositional property is an important aspect for the adequate development of any complex system. However, context-aware components are necessarily indeterminate and consequently lack the compositional property. In order to retain this compositional property we develop a model of computation (CAPN) that concentrates the context-dependency of an entity in a control stream. The autonomous processes in this model are flexible in the sense that they can asynchronously adapt to and cooperate with their milieu. We coined this type of operation "democratic processing".

Results

The study of developing complex communicating systems is a multi-something problem, where something can be anything, ranging from disciplinary development to objective optimisation or from input to output. We succeeded in structuring the multi-something problem in three themes and provide practical approaches and case studies for each of them. The three themes are communication, coordination, and composition.

For communication we rely on a multi-view representation of the system. Each view targeting specific concerns and conveying dedicated information.

For coordination we developed the Adaptive Resource Contracts (ARC) framework that supports flexible designs with non-functional aspects made explicit: distortion, capacity, and resource utilisation. ARC component are truly context-aware; they implement abstraction, adaption, and cooperation.

For the compositional property, ARC isolates the indeterminate behaviour of a component through a so-called oracle. Because of this isolation, the network retains its compositional property. Our Context-Aware Process Network (CAPN) model is an explicit model of computation that implements ARC concepts for the class of stream-based applications with occasional event handling.

One

Introduction

THIS dissertation presents a framework for mastering the complexity of communicating systems. The composition of a group of communicating systems is a system[1] again. This dissertation concentrates on systems that facilitate communication; a typical system has a backbone network infrastructure and communicating mobile terminals. The considered communication system is flexible in the sense that it 1) supports multiple types of media and 2) can be applied in different scenarios. For each scenario, the environment in which the system operates may vary and the expectations a user has of the behaviour and performance of the system may change. Developing a communication system under these constraints is a real challenge. It requires efficient coordination among the constituent subsystems. With the evolution of communication systems, users have become spoiled and expect increasing performance. As a result, systems require high-performance components, which complicates the development process even further. We propose a development framework that respects and values the potentials of individual subsystems and guides the coordination among components. An asset of our development framework is its applicability throughout the different stages of the life cycle of a system: during the exploration-time, design-time, compile-time, and run-time stages.

In this introductory chapter we set the context of our research and sketch the concepts of our development framework. The framework is the result of research conducted in the Ubicom research programme. Throughout this dissertation the Ubicom system, studied in this programme, serves as an example of a complex system for communication. The development framework is based on three key notions: abstract communication, context-aware coordination, and compositionality. It encourages the self-assessment of individual subsystems and it distributes the responsibility for the integrity of the system over the participating subsystems. Much of what you would expect to find in a true *democratic* framework.

[1] If necessary to discriminate between either two appearances of a system, we refer to the constituent systems as subsystems and to their composition as the compound system. A subsystems, thus is a component of a compound system

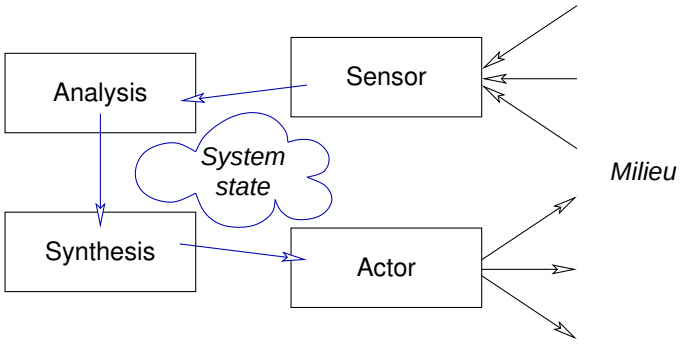


Figure 1.1: Generic communication system structure

1.1 Communication systems

Communication systems interact with their immediate environment, their *milieu*. A generic high-level structure of these interactions is given in Figure 1.1. The system depicted in this diagram is composed of four (sub)systems: Sensor, Analysis, Synthesis, and Actor. The sensor system attributes selected properties of the immediate environment; it senses the system context. The sensed information is transferred to the analysis system which effectively transforms the external context to the System state. The system state is shared by all subsystems involved and therefore contributes to the context of individual subsystems. The synthesis system takes the system state, which is now in the appropriate format, and prepares a proper (re)action of the system to its environment. The actor system performs the necessary transformation to a suitable output format. The composition of subsystems yields a system that interacts with its milieu, although none of the individual subsystems has this property; it is an emergent property of the communication system as a whole.

1.1.1 Ubicom

The Ubiquitous communication (Ubicom) research programme conducted at Delft University of Technology provided the inspiring context for the research described in this dissertation. The programme targeted visual augmented reality. To accomplish this target, the programme brought together a wide range of disciplines. To give an idea, experts from electrical engineering, computer science, physics, geodesy, and industrial design worked together on the Ubicom system. Ubicom pursued the development of a system for mobile visual augmented reality. A typical Ubicom application combines entertainment and information. The enterprise view of this *multidisciplinary* research programme is best introduced with an example. Example 1.1 presents a view on the Ubicom system from the perspective of the "interested" user.

Example 1.1 (Visual augmented reality) *The museum Boijmans van Beuningen is a multi-faceted museum with a representative collection of old masters of modern art [Boijmans Van*

Beuningen, 1849,1958]. Picture a visitor wearing the Ubicom terminal: a small device connected to a see-through display. The device has a wireless interface to the hidden backbone of the Ubicom system: ubiquitous computation and storage resources in the backbone network infrastructure. The system is context aware: it knows the position and orientation of the user as well as the user's preferences. The Ubicom system situates virtual graphical objects in overlay with real-world artefacts by projecting them on the see-through display such that, from the point of view of the user, the objects seem to be connected to real-world objects: visual augmented reality.

The Ubicom system executes a tour guide application. A small virtual creature acts as a guide who suggests possible interesting pieces of art. He leads the way to the old masters section. Here, the visitor enjoys a painting called "De kwakzalver (The Quack)" by Dou [1652]. The painting is an exemplar of the so-called Dutch genre paintings, which are full of symbolic references. When the visitor gazes at the painting, vivid animations clarify the scene pointing out the symbols, which refer to the contrast between luxury and austerity, or indicate dim-wittedness and deception. The visitor can alos choose to study the various hidden sketches which were the basis of the painting as it is today. These so-called underdrawings can otherwise only be seen separately using infra-red, ultrasound, or roentgenographic technology.

The new wing of the museum (author's imagination) accommodates the D.I.Y. hall. Here visitors are invited to sculpture and re-sculpture virtual objects of art. The room is equipped with empty frames mounted to the wall and socles placed on the floor. With digital paint and digital clay visitors make abstract works of art. The sculptures are left behind for others to enjoy or to modify. Particular successful collective works of art are conserved and put on display.

The leitmotif of this dissertation is the development of a Ubicom system as a *system*. The development does not take place in isolation; it is part of the Ubicom project, which in turn is part of the aforementioned Ubicom research programme. The Ubicom project addresses the technological issues of the research programme.

1.2 Complex systems

The development of a communication system in a multidisciplinary setting such as the Ubicom project is inherently difficult; it is a complex system in a complex setting. The complexity of any system is increased by the interplay of a number of aspects. The following are a few outstanding ones, which when used in combination define a complex system.

Scale Large-scale structures tend to obstruct a clear view.

Interaction Interactions among components may complicate their coordination.

Diversity The involvement of diverse technical domains (disciplines) induces heterogeneity in modelling and research approaches.

Irregularity Regular and homogeneous structures or interactions can be mastered by good bookkeeping alone. Irregularity and inhomogeneity complicate systems.

Flux The immediate environment of a system demands that the system is in a constant state of flux.

The Ubicom system is composed of subsystems that originate from various disciplines. Consequently, the Ubicom project had many, often competing, challenges [Lagendijk, 2000b]. Common in multidisciplinary projects is the tension between the individual and the collective. Researchers want to compete with international top-research institutes in their respective domain of expertise, while at the same time interdisciplinary collaboration is desired. The project-wide commitment is to explore collaboration and to have it result in a mobile augmented reality terminal, or even better, in a family of mobile augmented reality terminals; each terminal being optimised for a specific scenario.

The above challenges have in common that they target enabling technology rather than providing new applications or developing novel services. Several scenarios of possible application areas have passed in review during the exploratory stage of the life cycle of the system [van der Schaaf, 1999]. Each scenario describes an application and the *context* in which the system operates. The set of possible scenarios serves as a focus point and it enables high-level communication between disciplines. Defining a set of possible scenarios rather than a single scenario demands *flexibility* at all parts of the project. Flexibility is required when the individual system components are designed and flexibility is required when the system is assembled from these components. Proper coordination of the offered flexibility is constrained by the role a component plays in the entire system and the environment of the system. In short, systems must be aware of their context, irrespective of whether the system is a subsystem or a compound system.

In this dissertation we address the problem of developing complex communication systems. We focus on the proper coordination among the constituent components, because we belief that the integrity of a complex communication is significantly determined by its coordination. We will argue that a feasible method of coordination a ideally organised *distributively*, *non-iteratively*, and *evolutionary*. Since,1) Centralised solutions are impractical, if not infeasible, due to the complexity of the system under design. 2) Iterative solutions hinder timely designs due to the number of disciplines involved in the system. 3) Newly derived results must be seamlessly incorporated in the coordination framework. 4) Systems are in a constant state of flux.

In this dissertation we propose a democratic[2] and distributed approach to the coordination. Democratic because components must be aware of their context and acknowledge the fact that in turn they participate in the context of other components. Distributive because we want to support evolution and to capture expert knowledge. Because of the complexity of the systems under consideration a hierarchical organisation of the coordination is less favourable. Instead we advocate an heterarchic organisation that allows, in contrast to an hierarchical organisation, for a temporarily inversion of the dominance among communicating subsystems.

The heterarchic coordination is supported with the concept of abstract communication and compositionality. Abstract communication structures the system under development by taking different perspectives in order to separate concerns, whereas compositionality is essential for predicting the attributes of the compound system.

[2] Democracy [Procter, 2000]: "the belief in freedom and equality between people" in which we substitute people for subsystems.

1.3 Democratic processing

Our approach to master the complexity when developing a complex communication system yields a development framework that implements democratic processing. The development framework is organised around three themes: abstract communication, heterarchic coordination, and composition.

i. Complex systems have a wide range of properties that are hard to value in a single stroke. If one considers only a confined set of properties, the view of a complex system usually shows many components. Structuring the view with nested levels is a proven method to control the breadth of the view while keeping the possibility to appreciate the depth of the view. However, during the development of a complex system, one needs to change perspective frequently. Sometimes detailed aspects of the functional structure are important, whereas at other times implementation and non-functional details prevail. Maintaining a consistent view on the interdependencies of the two aspects is crucial for proper system development. Our development framework provides an *ontological* model that relates different views on the same complex system. The views communicate concerns in an abstract way.

ii. Conveying clear information is essential for the coordination of a communicating system. Communicating systems have inherently complex interaction patterns, partly because of the close collaboration among subsystems. The definition of clear information, therefore, is highly context dependent. The role and the application of a subsystem – being part of the system as a whole – determines its context. Our development framework includes a quality of service framework (ARC) that provides an *informational* model to exchange functional and non-functional aspects among subsystems. Common identifiers in this model are distortion, capacity, and resource utilisation. The information model instruments local optimisation methods for doing system-wide tradeoff analysis.

iii. Compositionality is an essential property of any non-trivial system; the ability of constructing complex systems from its constituents. From a functional only perspective this is a solved issue. However when taking into account non-functional aspects, compositionality is only guaranteed in under specific conditions; the subsystem is either determined, or it can capture its context dependent behaviour. Our development framework provides the latter. Each subsystem has an so-called *oracle* that captures its indeterminate part, thus retaining the compositional property.

1.3.1 Abstract communications

The observation that different views use different definitions but are phrased in similar terms is important. It is equally important to recognise that each view exists for a purpose. The mere recognition of the underpinning problem is the first step of solving it. We use Bunge's *systemic* philosophical position and corresponding ontological framework [Bunge, 1977, 1979] to interrelate multiple coexisting views on a system. Each

view addresses one or more *concerns* (aspects) as articulated by the respective *stakeholder*. Views are materialised through constructing a modelling language. Hence, a view is a conceivable artifact of the system properties.

When navigating through a system, the perspective changes frequently. Sometimes only functional properties are important, whereas at other times also non-functional properties are important. On tour, it is very likely to encounter homonyms. This need not pose a problem, since a given context resolves any ambiguity. Consequently, we argue that it is not necessary to develop a metaframework that contains every possible property of the system. Instead, it is sufficient to switch views when necessary. However, it is important to recognise a change of perspective.

In our view the system architect should be flexible in taking different perspectives of the same system, which implies a frequent change of language. The required level of completeness of each language depends on the purpose. Scope turns out be an important parameter. In a communicating system, for instance, neighbouring components require a rigid language for their information exchange, whereas remote components can suffice to use an informal language. In the Ubicom system, components located in the vicinity of the user need not have a clear understanding of the communication patterns in transceiver layers. A classical example from the Ubicom programme is whether or not to bother the user with the current value of the so-called τ_{rms} or, vice versa, whether or not to install a switch in the transceiver layers to notify the fact that the user arrives at level 4 in his adventure game. Hiding the irrelevant properties (*abstraction*) is a useful tool to convey the essentials.

Similar observations have been made in various other disciplines. From the field of artificial intelligence Gruber, for instance, observes that clear communication between connecting agents is a prerequisite, but communication between remotely connected agents can be less rigid [Gruber, 1993].

In software engineering, aspect-oriented or generative programming is an emerging discipline [see e.g. Kiczales et al., 2001; Lieberherr et al., 2001; Ossher and Tarr, 2001; Kiczales et al., 1997; Czarnecki and Eisenecker, 2000]. Lieberherr et al. advocate the "Law of Demeter", a set of guidelines that organises and reduces dependencies among components. The law is inspired by the motto: "Only talk to your immediate friends". In addition, this programming paradigm facilitates the design of a family of systems rather than a single system because of its ability to change aspects. To support multiple aspects, individual component must show the necessary flexibility.

The above consideration are summarised in the following postulate.

Postulate 1.1 (No exhaustive metaframework)
In order to master the complexity of a system one needs only to develop a set of interdependent views. It is not necessary to have a metaframework that disambiguates all concepts of the system.

1.3.2 Heterarchic coordination

There are many ways of designing, composing, and coordinating complex communication systems. The traditional – hierarchical – way is to roll out a structure of related system components and to assign an architectural team that specifies the context for each

system component. The specification is input to the individual system designer. Within this context the designer develops an implementation that is usually best-effort as well as worst-case. The design is worst-case because it is dimensioned such that the specified functionality and performance is achieved under the worst possible conditions. The design is best-effort because all available resources are thoughtlessly acclaimed. From the perspective of the individual subsystem developer such a design is quite an achievement, but from the perspective of the system as a whole, more consideration is required; best-effort is not good enough. Integration of best-effort subsystems yields an aggregate, in the sense that the whole is most likely not an optimal system. The problem stems from the fact that best-effort is defined from a dedicated perspective, which not necessarily coincides with the system-wide perspective.

A structured view on the system includes a nested level representation, which is typically considered a hierarchical representation, where higher levels dominate lower levels of the structure. Components are included in multiple views and thus participate in multiple hierarchical relations. Considering two system entities will show viewpoint-dependent dominance relations. In one view, entity A dominates entity B, whereas in a different view the dominance relations is reversed. Changing the dominance relation among subsystems is known as an *heterarchy* [McCulloch, 1945; Dilts et al., 1991; Hofstadter, 1985].

We recognise that a hierarchical, authoritative, and centralised design method has proven its practical value, but we think that we can and should do better especially for complex communication systems. For small-scale, homogeneous, and regular communicating (sub)systems, a hierarchical design approach is the preferred choice. To strive for optimality in more complex systems, one needs to use an appropriate design methodology that addresses a number of aspects. The following aspects relate to the observation that complex systems are in a constant state of flux.

Evolution During the course of the design of a system, new research and development results will arrive and should be considered for incorporation with the least possible effort.

Flexibility The universal system idea does not hold. Instead the design should support a family of systems each of which is suitable for a specific scenario.

Integrity It must be possible to value the integrity of a system family member with respect to alternative designs as well as to its context.

Hierarchical (authoritative) design methods usually lack adequate support for most of the above-mentioned aspects. One of the problems is their centralised coordination. In order to master the complexity, hierarchical design methods abstract subsystems by modelling their behaviour and performance in a fixed system-wide structure. The structure and models are imposed by a central authority: the architecture team. In effect the whole system is captured in a single (meta) framework. We object to this style of design as articulated in Postulate 1.1. New developments, research results, or new products put on the market will fit in as long as they do not trigger radical changes of the system-wide model. The impact of changing the system-wide model can be quite significant. Notorious are changes of the functional interface for a specific component.

Viewed from an aggregate level, the Ubicom system of Example 1.1 has multiple, often conflicting, objectives. The mobile terminal, for instance, must be small and energy efficient yet it should offer a vast amount of information and breathtaking graphics. A brute-force approach to solving the underlying *multiobjective* optimisation problem is infeasible, if only considering the tremendous amount of decision variables. Fortunately, the system is inherently structured and this structure can be exploited. Exploring the structure of the system implies a frequent change of the perspective to effectively narrow the scope of the view and, more importantly, to reduce the number of decision variables. The change of perspective partitions the initially large multiobjective optimisation problem in many small – but dependent – optimisations problems. These smaller optimisation problems can be solved more efficiently than the large one, if the large optimisation problem can be solved at all.

The development process of the Ubicom system inherently is a *multidisciplinary* process. Stringent objectives imposed on the system make close collaboration among subsystems, and therefore disciplines, unavoidable. The range of involved disciplines is rather broad. There are subsystems for position recovery, graphics rendering, human perception, and high-throughput wireless communication, to name just a few. At first sight, mandatory collaboration among experts from various disciplines complicates system development. But willing collaboration increases the integrity of the compound system: the whole is more than the sum of its parts. Collaboration paves the way for solving problems in the appropriate *domain* of expertise. A recurring issue is of course making a tradeoff and finding the balance between partitioning and solving the problem.

Figure 1.1 introduced the internal state of a system. The design of a system involves the distribution and the coordination of the internal state. This state provides the context of subsystems, yet at the same time the internal state depends on the behaviour and performance of the subsystems. A change in appearance of one of the subsystems thus affects the internal state and consequently changes the context of other subsystems. Adapting to a change of context may be tedious, which is particularly true for systems that involve many disciplines.

Our requirements for an appropriate design are summarised in the following postulate.

Postulate 1.2 (Design rationale)
In order to master the complexity of a complex multidisciplinary system, a design must obey two principles

i. No global control (structure);

ii. Non-iterative control (coordination).

Let us reflect on Postulate 1.2 and the previously mentioned list of requirements: system evolution, flexibility, and integrity. Evolution is key in competitive system design [Andrade and Fiadeiro, 2001]. Advances in technology alone are not sufficient to compete with alternative designs; Moore's law [Moore, 1965] predicts that the number of transistors per integrated circuit doubles every 18–24 months. First, new concepts at the circuitry level are necessary to put these transistors to use. Second, increase

of resources alone is not sufficient to enhance system performance. New constructs are a prerequisite to keep up with advancing technology and increasing demands.

Evolution of technology hits a philosophical topic. Different kinds of evolutionary processes exist. The classical Darwinistic evolution principle involves mutations, slight modifications of the characteristics of a component. Beneficial changes will remain, while others die out: the survival of the fittest. Interesting enough [Gould, 1977], Darwin never used the word *evolution* because of the implications of order. According to Darwin, there is no ranking of species. A more appropriate term of the theory known as Darwin's evolution theory would be "descend and modify" [Gould, 1977]; In Darwin's view, evolution is a random process which serves no purpose.

The term evolution is due to Spencer (1864) [Gould, 1977]. Spencer explicitly defined progress as the cooperation among internal and external forces: the system and its milieu. An evolutionary system is in a constant state of flux. Through continuous changes, a system effectively adapts to changing conditions and objectives of its environment. The single most important process in a system therefore is the coordination between subsystems. Gould refers to the coordination as "regulation": the decision and timing of control. Both aspects are crucial. The decision of how and when to use a subsystem determines the behaviour of the eventual system and its emergent properties. Thus, given that a system is in flux, a timely decision is as important as an optimal decision. Consequently, optimality is put in perspective. A suboptimal decision – from a mathematical point of view – that is yet a timely decision may yield a perceptively superior system than the other way around. So, a system in flux need not strive for first time right.

Coordinating complex systems without a global, omniscient, controlling entity is observed in daily life in numerous situations. Take for instance a city like New York [example due to Holland, 1992]. Although there is no central controlling entity, shops get stocked and citizens can usually do their daily shopping. In case of a calamity, the balance may be disturbed for a while, but the system is self-recovering. At governmental level we notice economists, politicians, and theologians trying to understand the larger picture in order to control the community. In practice, they rule at an abstract level. At best, governmental ruling facilitates the interaction between groups of people while breaking the dominance. The famous Dutch "poldermodel" became a classic example of breaking the hierarchy. The government no longer simply sets the rules, but opens up the dialogue with business and non-profit organisations to arrive at a consensus of good practice.

We refer to this type of distributed coordination as *democratic processing*. There is a bidirectional flow of arguments. History proves that this type of organisations can be very competitive. In Ubicom we have applied democratic processing throughout the life-cycle stages of the system. During the exploratory stage, informal dedicated groups gathered to exchange abstracted information. The contours of the context for individual subsystems resulted from these meetings. We developed a framework for negotiated quality of service (QoS) to capture and distribute the (inner) system context. The QoS framework is named ARC: adaptive resource contracts. The framework is applied during the design-time stage as well as during compile-time and run-time stages. ARC enhances the system context by making relations explicit. Explicit relations combined with context-aware subsystems are the basis for a fruitful design.

1.3.3 Compositionality

A proven method to develop complex systems is based on decomposition, e.g., applying a divide and conquer strategy. However in many practical situations the final phase of system integration fails because of a false assumption that the set of system components is indeed compositional.

One source for this false assumption is due to the limited number of system aspects that are considered. For a complex system it is insufficient to consider only functional aspects. Non-functional aspects such as capacity and resource utilisation are of equal importance.

Another source of the wrong assumption is due to a limited awareness of the system context. Only in the rare practical case that components are *determinate* their compound is also determinate, hence *compositional*. In general the behaviour of an indeterminate component depends on the context it is applied in. As an example consider a real-time system that executes a set of tasks in a time-shared fashion. Tasks are assigned a priority, which assists the system to find a schedule that can cope with the individual real-time constraints of each task. At first sight, tasks seem to have a determinate resource utilisation, e.g., CPU usage. However switching tasks consumes resources as well and moreover the amount of resource utilisation of a task switch depends on the current state of the task. The seemingly determinate task turns out to be an indeterminate one after all and consequently the behaviour of the system becomes schedule dependent.

Compositionality is a necessity for developing complex systems. In this dissertation we implement two strategies, captured in the following postulate.

Postulate 1.3 (Compositionality)
The following two strategies can achieve compositionality:

i. The interface among cooperating components explicitly addresses non-functional aspects such as capacity and resource utilisation.

ii. System components are truly context-aware. Any indeterminate system aspect is made explicit, which allows for a determinate composition of system components.

1.4 System Architect

The development of complex systems requires architecting and engineering. In [Rechtin and Maier, 1997], architecting is considered an art, whereas engineering is considered a discipline. The problem of proper coordination among system components is closely interwoven with the responsibilities of a system architect. This dissertation takes the perspective of a system architect.

It has been acknowledged that a universal recipe for development of complex systems does not exist. Rechtin and Maier give an exhaustive list of recipes and points of particular interest [Rechtin and Maier, 1997]. Common to all methods is a *hierarchy* of system architecting entities. At the top, the system architect, a generalist, dominates

a range of *aspect* architects [Muller, 2001]. Aspect architects in turn use other aspect architects at a lower level. At the bottom of the hierarchy is the specialist, who is an expert in his domain. The above view is shared with the upcoming model-driven architecture (MDA) of the object management group (OMG). MDA aims at connecting enterprise (business) modelling with technology. By concentrating on modelling rather than implementations MDA may combine common off-the-shelf components. In [Bézivin and Gerbé, 2001], four principle levels of modelling are determined: meta-meta (corresponds with the system architect), meta (corresponds with the aspect architect), model (corresponds with the specialist), and implementation.

Our approach is different in the sense that we acknowledge aspect architects but do not have a dominant system architect. In our philosophy the system architect is the *primus inter pares*. At best the system architect mediates collaboration among components and component owners. Since coordination of subsystems is the main topic of this dissertation, we regard the system architect as a catalyst. In a recent column, Fowler coined this type of architect the *Architectus Oryzus* as opposed to the *Architectus Reloadus* who is the person that makes all the important decisions [Fowler, 2003]. We consider cooperation (heterarchy) an important aspect of our approach, which is lacking in the above hierarchical structure. The main design flow of MDA is top down. A bottom-up flow is forcefully introduced through reverse engineering.

1.5 Outline

This dissertation presents a development framework based on three themes: abstract communication, heterarchic coordination and composition. Where communication and composition are supporting themes for the heterarchic coordination.

In Chapter 2 (Complex systems in perspective) we address the topic of abstract communications. In this chapter we discuss an ontology of the world that effectively structures the concept of views and their relations. We deploy a recent initiative for the development of complex systems: the IEEE-1471 standard [Maier et al., 2001].

In Chapters 3 and 4 we address the topic of heterarchic coordination. In Chapter 3 (Negotiated Quality of Service) we present the general concepts of this framework, in Chapter 4 (Mathematical consideration of QoS) we consider the underlying multiobjective optimisation problem. ARC implements: abstraction, adaptation, and cooperation. It conceptually distributes the coordination, yet leaves room for an implementation with a global controller. ARC facilitates non-iterative control, it uses multiple objectives as a tool to enlarge the scope of individual subsystems. Instead of having a single point of operation, subsystems offer multiple non-inferior (or non-dominating or Pareto) points. The net effect is that system components are aware of their context. Awareness works both ways. Components *adapt* (inward) to changes of their environment, but also *cooperate* (outbound) with their environment. We have previously reported on ARC and its applications in [van Dijk et al., 2000b,a; Taal et al., 2002].

In Chapter 5 (Context-aware process networks) we address the topic of composition. In this chapter we develop a model of computation: context aware process networks (CAPN). It introduces cooperation in a stream-based model that makes components context aware. The CAPN model retains the compositional property for networks

that are inherently indeterminate. Although CAPN is a specific instantiation of ARC, it demonstrates that systems that implement ARC, capture their context-dependent behaviour and consequently have the composition property. A previous publication of the CAPN model of computation can be found in [van Dijk et al., 2003].

In Chapter 6 (Ubicom case studies) we presents several Ubicom case studies to demonstrate various aspects that have been developed in the abstract in Chapter 3. In this chapter we quantify two important qualifications of a system that uses the ARC framework: agility and efficacy. Finally, Chapter 7 concludes this dissertation.

Two

Complex systems in perspective

SYSTEMS and their associated architectures are often ill-defined notions, if defined at all. Individuals, however, have proper intuitions about their view on a system and its architecture. But more often than not, intuitions of individual stakeholders turn out to be incompatible, which seriously hampers clear communication of concepts. In this chapter we discuss a number of views on a system. A subset of these views is demonstrated by using the Ubicom system as an example. All views are related to the systemic framework developed in [Bunge, 1979], which proved a useful reasoning framework for relating views and clarifying positions of domain experts. the chapter is concluded with discussion on the aspects that are necessary for a proper coordinating system.

In the development process of complex systems that involve multiple disciplines, incompatible intuitions are bound to happen. In part the incompatibility is due to different concerns of different stakeholders. In a multidisciplinary environment it is of the utmost importance to convey concepts in a proper and unambiguous way. Especially since the integrity of the eventual system is the aim of the design process.

Recent literature recognises the discrepancy between various intuitive understandings of concepts. IEEE and ISO have jointly developed standards, [IEEE-1471, 2000] and [RM-ODP-10746, 1998] respectively, to address this very issue. Both standards emphasise the existence of multiple views on the same property of a system: its architecture. The standards acknowledge that it is neither possible nor necessary to describe all possible details of an architecture in a single stroke. A more suitable method is to develop multiple *interdependent* views of the architecture. A view addresses a limited set of system aspects. A view articulates and values a selected set of system aspects through a so-called modelling language; this viewpoint model is a substantial artifact. The interdependency of the various views on the system is important for the integrity of the system. In this dissertation we apply the paradigm of interdependent views on system development, Muller describes a multi-view method that applies the paradigm to the entire life cycle of a system [Muller, 2004].

The various views on a system are linked, the rationale of these links are import-

ant for maintaining the consistency of the system. An interesting approach to identify the interdependence between viewpoint languages is the development of an ontology. Given a proper ontology, one can study the mapping of modelling language constructs to ontological constructs. By comparing these maps for different views one can discover their interdependency. An analysis of the map itself may reveal possible discrepancies of the modelling language [Wand and Weber, 1990].

We adopt the systemic position [Bunge, 1979]: an ontological view on systems. The ontology developed by Bunge serves as a frame of reference for further discussion. Bunge's ontology puts into perspective the existence of multiple views of the same system. Also, it is a reference framework to qualify individual views and their applied viewpoint language. In this chapter, we use the Ubicom system to exemplify a complex system which is described by multiple views. The terminology for the respective views is adopted from the ISO standard: "Information technology Open Distributed Processing reference model" [RM-ODP-10746, 1998].

2.1 Ontology

Ontology is and has been the subject of many philosophical debates. Ontologists seek to produce a unified picture of reality: the concrete world. In this section, we discuss an ontological framework that will be the reference for relating various views on abstract – beyond our reach – properties (or facts) of a system.

Bunge wrote an extensive treatise on basic philosophy. In [Bunge, 1977, 1979] he develops the systemic position, "Systemism encourages attempts to analyze systems into their composition, environment, and structure as well as to disclose the mechanism of their formation and breakdown" [Bunge, 1979]. Systemism is a balance between *holism* and *atomism*. Holism [see e.g. Healey, 1999] is the belief that every natural thing is connected to every other thing; the thing is a whole, which is more than the sum of its parts. Atomism, on the other hand, [see e.g. Irvine, 2001] regards the world as consisting of a complex of atoms and their properties. When combined, these atomic facts form complex compound objects.

Bunge derived his systemism ontology in a formal and mathematical way [Bunge, 1977]. Here we present an informal version of the model. Subsequent sections put in perspective the IEEE-1471 standard and the modelling language evaluation method of Wand and Weber. The recent "Recommended practice for architectural description of software-intensive systems" [IEEE-1471, 2000; Maier et al., 2001] recognises the mere existence of multiple frames of reference in the development of complex systems. This position coincides with the systemic position. Wand and Weber have adopted Bunge's ontology to form an ontological framework for modelling language constructs. Their so-called Bunge-Wand-Weber (BWW) model serves as a reference framework that allows a conceptual analysis of several modelling languages.

2.1.1 Systemism, an informal introduction

Ontology tries to formalise a perception of the real world. A frequently quoted definition is that of Gruber: "An ontology is an explicit specification of a conceptualization"

[Gruber, 1993]. Bunge's systemism considers a *universe* that is inhabited with *things*. In Bunge's ontology, things are individuals, substantial or conceptual, with corresponding *properties*. Properties (or facts) are generally beyond our reach, *predication* makes a property conceivable; value the property of a thing from a dedicated frame of reference. A predicated property is called an *attribute*; a predefined reference frame (point of view) is called a *manifold*. The manifold can be regarded an impartial but biased spectator; it has dedicated concern and therefore often referred to as the *stakeholder*.

Example 2.1 (Manifold) *Consider a pair of dice: white-coloured dice with black spots (a fact). An observer who wears a pair of red-glassed spectacles will predicate the pair of dice as being red. Dice are used for stochastic games. The ordinary interpretation (predefined manifold) is to throw the dice and to count all top-view visible spots. For a pair of perfect dice, the consecutive throws are stochastically independent. Each throw has a uniform probability mass function* $(1 \cdots 6)$*. The stochastic aspects of a series of throws of the pair has a triangular probability mass function* $(2 \cdots 12)$*. Other games may have different interpretations of the top-view visible spots. In a game of poker, for instance, structure is the primary attribute and value a secondary one. One can even go as far and make the interpretation itself the object of the game.*

Different individuals (things) are related when they have mutual properties. Things may have *bonding* (link) or *non-bonding* (mere) relations. A bonding relation implies action, and possibly re-action of the things involved in the relation. The set of bonding relations of a set of things is called their *bondage*. A non-bonding relation predicates the relation itself; thing A is larger than thing B. The set of non-bonding relations of a set of things constitutes their *configuration*.

The *state* of a thing is the aggregate of its predicates and henceforth viewpoint dependent. An *event* (action/re-action) induces a state transition; a *process* is a state transition trajectory: the course of events. The history (evolution) of a thing captures the state transition trajectory. When two neighbouring things influence their respective evolution of their state transition trajectory, these things have a bonding relation, otherwise their relation is non-bonding.

A compound thing is composed out of two or more things. A compound thing is qualified as a *system* if the bondage of its constituents is non empty. Because of the bondage, a system has emerging properties. For a bondage implies the invocation of events. A compound thing of which the constituents only have mere (non-bonding) relations is an *aggregate*. The environment of a system habitats things. Usually only the immediate environment, or *milieu*, is of direct concern. The milieu constitutes those things outside the system that participate in the bondage of the system under observation. Note that the bondage is manifold dependent, for the bonding relation links properties of individual things.

Example 2.2 (Milieu) *Recall the pair of dice of* Example 2.1*. Let the ambient lightening be red stained. Now, the dice appear to be red coloured even when watched without the specially prepared spectacles. Suppose we add a third, somewhat larger die. The third die has a non-bonding relation with each die of the primary pair: it is larger than each die of the pair. The third die happens to be different in another aspect as well; The third die is a magnet, whereas the pair is made of soft iron. Obviously, the stochastic properties of the individual throws of the pair of dice change when the third dice enters the immediate neighbourhood (milieu) of the pair; a throw of one die is no longer independent of that of the other dice.*

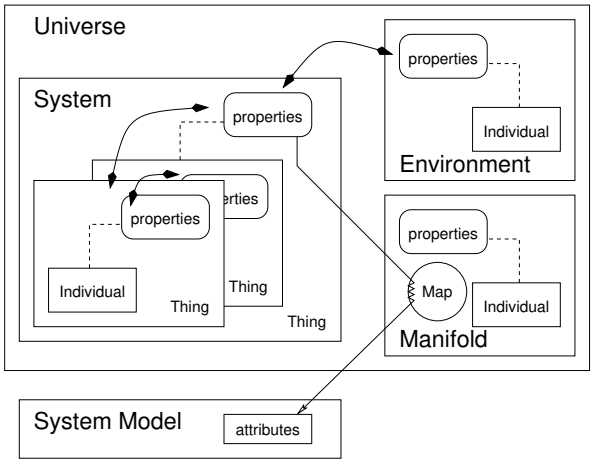


Figure 2.1: Systemic frame work.

Figure 2.1 is a graphical interpretation of Bunge's systemic position. The attribution of properties is *context* dependent. First, the environment of the system influences the system. The immediate environment (milieu) inhabits properties that relate to system properties. Second, the frame of reference (the manifold) determines the eventual value assigned to a selected property. A system may be perfect in one scenario, while useless when applied in a different scenario. But also, a system may be important from one perspective (making money) while considered a trifle from, say, a technical point of view. Given the ontological framework, the system context has a precise definition: the context of a system is the attribution of environmental properties that are part of the bondage of the system. The actual assigned value to a selected property depends on the manifold.

Figure 2.1 visualises the fact that the systemic position recognises different views of the same system without being discriminative. The manifold manifests the take perspective; the concern by which the system is addressed. Also, Figure 2.1 emphasises the fact that the context of a system plays an important role in the evaluation of the performance and behaviour of the system.

A principal predicate of a system is the *integrity* of a system. A related notion is the *cohesion* of a system: the degree of integration of the components of which the system is composed. Two objects are coordinated when they jointly contribute to the integrity of the system. Whether or not the coordination requires intervention of an external controller is left unspecified. Integration and coordination are two separate notions. When integration fails, structural breakdown is the result; when coordination fails, functional breakdown is the result. The overall goal of the system architect is to hit the structure that maximises the system integrity. We conclude with a postulate [Postulate 1.8 Bunge, 1979]

> The more cohesive each subsystem the less cohesive the total system.

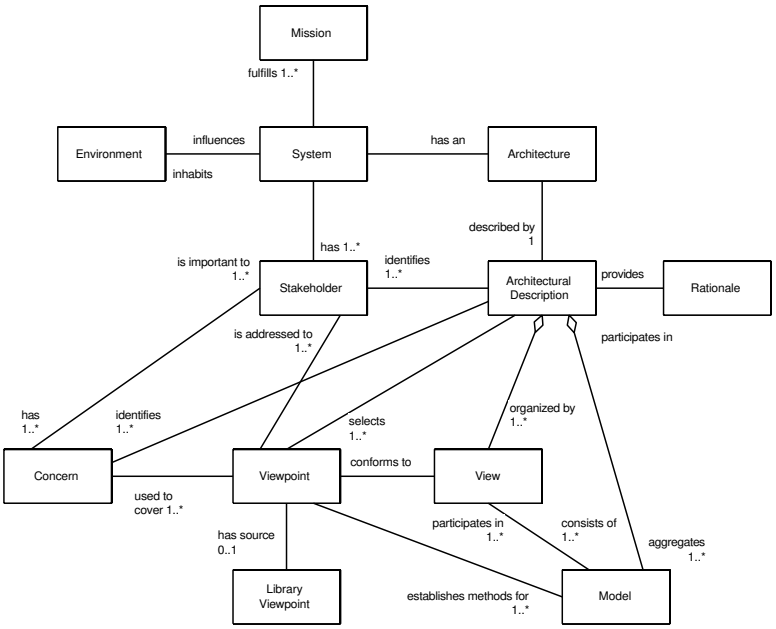


Figure 2.2: Conceptual model of the recommended practice for architectural description [IEEE-1471, 2000]

A cohesive total system implies rigidity, whereas we strive for flexibility. Following Bunge we should aim at cohesively developed subsystems.

2.1.2 Recommended practice for architectural description

The growing awareness that complex systems cannot be described in a single view triggered people to rethink their practice of architecting such systems. One example is the recently emerging IEEE-1471 standard. The standard is consistent with Bunge's systemic position. The standard is best explained by Figure 2.2 [IEEE-1471, 2000, Figure 1].

Figure 2.2 is in fact an instantiation of Figure 2.1. The system entity is the subject of Figure 2.2. A system fulfils a mission and it serves a purpose. The system context is important since it determines the properties of the system. The environment and the system are linked through their mutual properties. The manifold defines the attribution of properties. The system architecture is a principal property of the system. The architectural description (AD) of Figure 2.2 corresponds to the manifold of Figure 2.1. The AD assembles the possible viewpoint and the concerns of selected stakeholders into a frame of reference. The AD attributes the architecture property into a view: a conceivable artifact. The standard manifests the syntax and semantics of the modelling (viewpoint) language; the model entity of Figure 2.2.

IEEE-1471 recognises that the architecture is an important property of a system. In [IEEE-1471, 2000], an architecture is formally defined 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". The informal definition from the accompanying frequently asked questions sheet is even more close to our observation that it is a property that is generally beyond our reach, but tacitly available: "So, what is an architecture? We're not sure, but we know one when we see one".

The appendix of [IEEE-1471, 2000] refers to [RM-ODP-10746, 1998], a recent reference model for open distributed processing. This ISO standard describes in detail the concerns, the stakeholders, and the applied modelling languages. The disciplines in Ubicom have a broader scope than distributed processing, however; the viewpoints developed in [RM-ODP-10746, 1998] offer a suitable starting point for an accurate description of the Ubicom system. We adopt their terminology, but with a slightly different interpretation. Each of the views considers the scope of a limited set of system properties.

Enterprise view A high-level view from the perspective of users and owners. The enterprise view concerns the coordination among disciplines and their requirements for flexibility and adaptation.

Informational view A view on the semantics of information. The informational view concerns data, context, and operational information.

Computational view A view on the structure of information processing. The computational view concerns policies to support distributed implementations, e.g., for offloading computational intensive tasks.

Engineering view A view on the mechanisms and functions to support distributed implementations.

Technical view A view on incorporated standards and applied technology.

The identification of views is important for two reasons: the views help to maintain the consistency of the system under development and the views help to structure the necessary coordination among subsystems. In Section 2.2 we present the identified views, taking the Ubicom system as a representative example. Relations between views are important. Making relations explicit allows individual stakeholders to navigate different aspects of the system, while exploiting the benefits of maintaining a limited scope. In Chapter 3 we use the above views to develop a framework for coordinating subsystems.

2.1.3 Bunge-Wand-Weber and modelling languages

Wand and Weber used Bunge's work to construct an ontological framework to compare the expressiveness of modelling languages. Their framework is known as the Bunge-Wand-Weber (BWW) model [Wand and Weber, 1990, 1995; Wand et al., 1999]. The BWW model acts as a reference framework for modelling languages. It is a suitable

instrument to predicate (qualify) the accuracy of a modelling language. The basic idea is to evaluate the mapping of constructs from the modelling language under observation to corresponding constructs of the BWW model.

Modelling languages must strive for *semantic disambiguation*. Recall that a view is a conceivable artifact expressed in an appropriate modelling language. A proper view thus conveys clear information. A language construct can only convey clear information if it has a unique semantical interpretation. A true ontology is unambiguous; the BWW model is conjectured to be a true, possibly incomplete, ontology. Modelling languages, with respect to the BWW model, thus must be free of [Wand et al., 1999]:

Construct overload A single modelling construct representing two or more ontological constructs (*N* to 1 map).

Construct redundancy Two or more modelling constructs representing the same ontological construct (1 to *N* map).

Other publications [Green and Rosemann, 2000; Wand and Weber, 1995], in addition impose avoidance of two less destructive properties that lack a one-to-one relationship between the modelling and the ontological constructs: *construct excess* and *construct deficit*. Construct excess indicates modelling constructs exist with no equivalent ontological construct, while construct deficit indicates the opposite. These two discrepancies hamper completeness, but do not jeopardise precise conveying of information.

Various researchers have evaluated a wide range of modelling languages with respect to the BWW ontological model. Green and Rosemann analysed the integrated process modelling [Green and Rosemann, 2000], Wand et al. themselves analysed the entity-relationship model, and [van der Aalst, 1999] evaluated the event-driven process chain model. Process models, in particular the event-driven process chain model, play an important role in enterprise modelling, whereas entity-relationship models are important in information system modelling. A striking result of the aforementioned research is that all evaluated models have discrepancies, yet reputable design methodologies have been built on these modelling languages. Construct overload is detected in all cases, most notably in the event-driven process chain model, which is the basis of SAP R/3 and ARIS, leading products in the field of work flow management and business process re-engineering [van der Aalst, 1999]. Green and Rosemann place a critical note: whether or not the observed discrepancy is a deficiency of the ontological model or a deficiency of evaluated modelling grammar is open for debate [Green and Rosemann, 2000].

Recently, the BWW model has been applied in object-oriented designs [Opdahl and Henderson-Sellers, 2001]. Opdahl and Henderson-Sellers use a so-called facet modelling language for this purpose. The modelling constructs of this language are interpreted and analysed in relation to the BWW model. Possible semantic deficiencies are resolved by reformulating the facet language constructs. Hence an unambiguous modelling language remains.

The BWW model provides a suitable tool to analyse the modelling language of a viewpoint. In this dissertation however we choose not to take this formal direction. Ambiguities are resolved through close cooperation among subsystem developers.

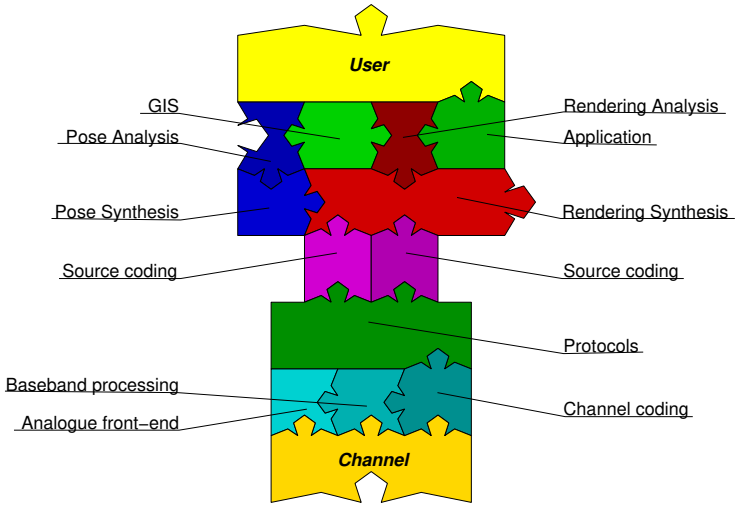


Figure 2.3: Conceptual model of the Ubicom mobile terminal system.

2.2 Ubicom

In Section 2.1 we presented an ontological view on complex systems. In this section we use the Ubicom system to illustrate the introduced views on complex systems. We claim that the Ubicom system is sufficiently generic to make the views an integral part of our framework. The views presented in this section do not yield an exhaustive description of the Ubicom system. Emphasis is on concepts rather than on details. The views are interdependent; when appropriate we will emphasise these relations, whereas at other times we will use transparency to hide irrelevant details.

2.2.1 Enterprise view

The enterprise view is a high-level view on the Ubicom system. It represents the Ubicom system from the broad perspective of owners, user groups, and designer collectives. A general introduction on the Ubicom system and as such part of the enterprise view is given in Example 1.1 on Page 2.

An abstract illustration of the Ubicom system is the jigsaw diagram of Figure 2.3. The diagram represents a functional/conceptual model of a mobile terminal. The multidisciplinary character is indicated by the disciplines associated with specific system components. The system context is attributed by the user objectives and the channel conditions. Possible objectives are: a long operation time with a perceptively poor quality, or a short operation time with a perceptively good quality. An example of possible conditions is (assuming a fixed throughput): low transmit power and low signal-to-noise ratio (SNR) or high transmit power and high SNR.

Figure 2.3 has two orthogonal flows. The application flow from left to right and the support flow (for communication and control) from bottom to top.

The application flow (c.f. Figure 1.1 on Page 2) starts at the left-hand side with a positioning component that uses sensors, whose output is analysed to determine the position and orientation of the user. Subsequently, the application drives a 3D graphics synthesis component, which prepares graphical representation of the intended reply, whose perspective is correct from the viewpoint of the user. A see-through display outputs the virtual graphics in overlay and aligned with real-world objects. The see-through display is on the right-hand side of the diagram.

The support flow implements communication and control and supports the application flow. The communication flow allows for offloading of computation and storage-intensive tasks and it facilitates retrieval of causal information from a backbone support system. The control flow incorporates the human user in the loop. In order not to complicate the system description unnecessarily we assume the availability of infinite resources at the backbone infrastructure. Also we will not address multi-user communication explicitly. Fluctuating channel conditions may hamper timely availability of resource capacity at the backbone. At the top of the diagram the user is in control. He or she finalises the constraints on the overall quality of the provided service and the limitation of resource utilisation of the terminal application. The constraints reflect the objectives as specified by the user.

The enterprise view necessarily employs an *informal* modelling language. The enterprise view fulfils the role of an intermediate among a wide range of disciplines and consequently offers a broad view on the system. The main concerns addressed in the enterprise view include

- Demonstration of mobile augmented reality.
- Distributive organisation of the system.
- Ability of the system to adapt to context changes; changes of user objectives and variations of environment conditions.
- Coordination among components that originate from a wide range of disciplines.
- Efficient implementations of a Ubicom mobile terminal: low power and small form factor.

Mobile augmented reality has been introduced in Example 1.1 on Page 2. The example also introduces a distributed implementation. Distribution of functionality is necessary because of the foreseen computation-intensive and storage-intensive applications and services. It is generally conjectured that technology will never be advanced enough to implement all possible services on a mobile device given the current energy consumption and form factor constraints. The minute we resolve one issue, another one will pop up. The exponential increase of demands and expectations of a communication system is a good indication of the validity of this postulate. Offloading of services to a backbone system is an obvious solution. Another important reason for opting for a distributive approach is the causal relation between the generation and presentation of information. It simply might not be possible to assemble a mobile system carrying all necessary information, because such as system has not been generated yet (c.f. the D.I.Y. hall of Example 1.1).

The Ubicom system is a *true* system, not a mere aggregation of components. Being a system, it must have emergent properties; the whole is more than the sum of its parts. The most important emergent property of the Ubicom system is being *context aware*; it is aware of its relation with its immediate environment (milieu). The system context consists of varying user *objectives* and changing environment *conditions*. User objectives change from one application to the other, or better, from one application mode to the other. Environment conditions can be highly fluctuating. The distributive organisation implies the application of a wireless link between a backbone support system and the mobile terminal. Wireless links are notorious for their quality fluctuations.

Energy is the fundamental, yet scarce, resource in a mobile terminal. The use of storage, computation, and communication resources draws energy from the battery. The distribution of the energy budget over the various components in the system is crucial for the eventual *performance* of the system. In case of a context-aware system, thus with emergent properties, the distribution is not a static one. The assignment of energy to a component and the consequent contribution of the component to the system integrity depends on the system context and the role of the component in the structure of the system, i.e., the component context. There is a clear role for coordination to effectuate the assignment of energy.

Energy-efficient implementations require a malleable system assembly, at compile time and preferably also at run time. At compile time the appropriate version of a component must be selected. At run time the distribution of energy of the components must be coordinated. The jigsaw diagram of Figure 2.3 therefore is not a materialised one but a malleable one; the outline of the jigsaw as well as that of the individual pieces change with the objectives, conditions, composition, and structure of the system. In the ideal case, the jigsaw has closely fitting pieces without any gaps. This can only be accomplished with careful coordination among components. Because of the interaction of neighbouring components in the system, we may regard them as individual components with overlapping environments when pulled apart, figuratively speaking. One component inhabits the context of the other component and vice versa. In Chapter 3 we will pursue this concept further.

Although not in all possible detail, the enterprise view reveals part of the computation structure of the system. Neighbouring components in Figure 2.3 will interact, for better or for worse. Being neighbours indicates collaborative design of1) the compound functionality and 2) the interfacing between the components . The enterprise view is vague enough to indicate the role of a component in the system and consequently the view indicates high-level consequences of changing system objectives and varying system conditions. The enterprise view mediates in the collaboration between designers from a wide range of disciplines.

2.2.2 Informational view

The informational view concentrates on the semantics of the information flow. The view is particularly concerned with the information exchange between neighbouring components. We consider local communications more important than global ones. First, because effective global communications require a global (or meta) modelling

language of a viewpoint, which contradicts Postulate 1.1 on Page 6. Second, since neighbouring components inhabit each others milieu, collaboration is effective here.

Figure 2.3 loosely introduced the component structure of the Ubicom system. In this section we concentrate on the application flow of the Ubicom system. The support flow is described in more detail in Section 2.2.3. The application flow is a *filter*, from a sensor device through an analysis and synthesis phase to an actor device. The filter *adapts* to changing objectives and varying conditions.

The diagram of Figure 2.4 presents a high-level view of the application flow. At the top, a user interface (userIface) captures the user preferences and translates them into objectives for the application manager (applicationMgr). Concurrently a positioning (positioning) component determines the whereabouts [Persa and Jonker, 2000] of the system. The application manager uses approximate location information, whereas the graphics processor uses more accurate position information and also orientation information (graphicsProc).

The aforementioned application manager combines user preferences and user location information to generate a so-called scene graph (SCENEGRAPH). The scene graph is a construct which describes 3D graphical objects and their relations to each other as well as their relation to real-world coordinates. The real-world coordinates are taken from the positioning component. A scene graph can, for instance, position virtual graphical objects in the real-world connected to real-world objects: a virtual sculpture on a real pedestal (Example 1.1). The scene graph is the common construct, with a defined interpretation for communication between the application manager and the graphics system. The scene graph provides an informational view on the generated reply of the application.

The graphics system takes the position and orientation of the user as a starting point for rendering the scene graph. The resulting rendered view is a 2D map of the scene graph ready for display (display). Depending on the applied see-through display technique, some filtering (processing) is necessary in order to properly overlay the virtual world on the real world. An example of such a filter is matching the field of view of the display with the provided 2D map [Dijkhoff et al., 2000]. The structure of the filter is an aspect of the computational view. The performance and resource utilisation of the filter are aspects of the informational view. These latter aspects are exposed through the RENDEREDVIEW construct.

Graphical objects often have texture, still image, or video textures. The graphics system uses a source transformer (sourceCodec) to acquire textures (VIDEO) information in the appropriate format. From the perspective of the graphics processing part it is irrelevant whether the source transformer uses a pre-recorded or live video source.

The main concerns addressed in the informational view include:

- The semantics of the information flow between consecutive components.
- Specification of the immediate environment; conditions and objectives.

We address these concerns using three universally applicable selecting mechanisms: abstraction, adaptation, and transparency.

Abstraction combines and organises the relevant properties to convey information about components in a clear and well-defined way. The scene graph object for instance,

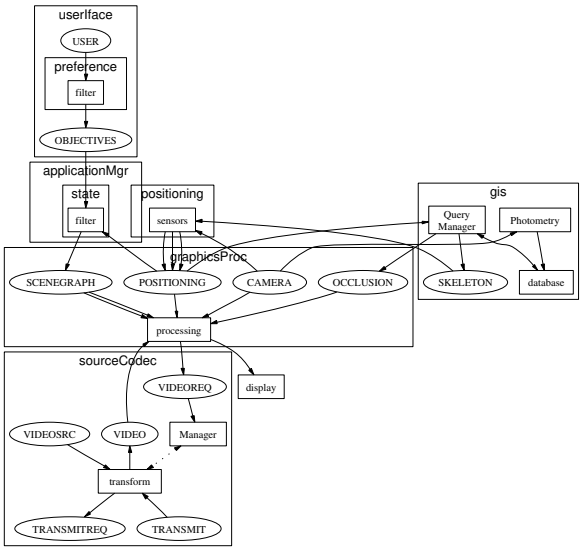


Figure 2.4: Ubicom application flow.

(SCENEGRAPH), models the virtual world by using language constructs and semantics. Although the application manager and the graphics system components may attribute different predicates to properties of the scene graph, they share a mutual understanding about a predefined set of properties. From the perspective of the graphics system, the nature of the process of *how* to derive a particular scene graph is irrelevant. However, the intentions of the scene graph must be clear. Vice versa, the application manager is not interested in the details of *how* graphical objects are eventually offered to the user, but only in maintaining the proper intention of their shape and relation. As an example consider a message to be conveyed to the user: "Battery low, immediate action required.". The graphics system must understand the priority of this message. Whether it chooses to use speech synthesis or a text display with contrasting colours is irrelevant to the application manager.

Adaptation is the natural process to react to a changing context, because practical components never operate in isolation. Capturing all possible states (the context boundaries) of the environment in a so-called *scenario* is usually not particularly useful, if at all possible. This is especially the case with the human user in the loop. A more practical approach is to allow for context changes, let the component observe them, and let the system adapt its mode of operation consequently. This way, the composition yields a flexible system that can be applied in a wide range of scenarios without the need for an explicit (and precise) definition of every possible state of the environment.

Transparency hides structural details and properties of a component. In this informational view we are concerned with the informational constructs (SCENEGRAPH, POSITION) rather than any processing construct (filter). The computational view (next

section) reveals more details of the processing structures.

The SCENEGRAPH object specifies graphical objects and their relations in a dialect of the virtual reality modelling language (VRML), which is a technological aspect. VRML [VRML-97, 1997] is an universally accepted modelling language with suitable constructs to clearly convey information. The POSITION construct describes the position in worldly coordinates and orientation in hemisphere coordinates. Accuracy of the offered coordinates is an important aspect of the POSITION construct. The attribution of accuracy though, is highly context dependent. A scenario that assumes a slowly moving user allows for less frequent updates of the positioning coordinates than a scenario in which users are very active, e.g., playing games. With respect to Example 1.1 on Page 2, the accuracy of the user coordinates must be precise enough to be able to overlay virtual graphics with real-world objects. Other applications may be suited with approximate coordinates, e.g., the room the user is in. Accuracy itself thus has multiple aspects: space and time. The discussion demonstrates that implementation matters. The informational view necessarily contains computational and engineering aspects. Depending the computational structure the impact of more accurate or less accurate coordinates is significant. Depending on the engineered organisation, coordinates become available in time or lag behind. Both aspects have their influence on the integrity of the system as a whole. In Section 2.2.5 (Technical view) we will come back to the applied standards in the Ubicom system.

2.2.3 Computational view

The computational view concentrates on the structure of how information and data is processed. It corresponds to a functional decomposition of the system. The concerns addressed in this view include

- The distribution of functionality over the disciplines involved in the system.
- The flow of information and data.
- Adaptation to a changing environment: conditions and objectives.

In Figure 2.5, the source coder receives a request for providing a video texture. The specifics of the request are specified through an informational construct: VIDEOREQ. The general manager of the source coder handles the request. It outsources the transformation, from a video source format (VIDEOSRC) to the specified video format (VIDEO). The transformation system which performs the actual transformation is generally composed of an encoder and decoder; the encoder translates the video source into an intermediate format, while the decoder translates the intermediate format to the externally requested texture format (VIDEO). Typically, the path from encoder to decoder includes a transmission system. In case of the Ubicom system, the transmission system is a wireless communication system that implements the support flow, see Figure 2.7.

The computational view of the graphics system (Figure 2.6) shows similar concepts to the view of the source coder. An analyser takes the incoming scene graph (SCENEGRAPH) and attributes properties to selected objects in the graph. Based on this analysis, the subsequent compilation phase synthesises a description of the scene graph

in an intermediate format. The compiler description is targeted towards the eventual thin-client 3D rendering [Pasman and Jansen, 2002]. The basics of the compiler description is a decomposition of the scene graph in polygon objects (frame description) and animated textures. The polygon description (ASSEMBLEDSG) and textures (VIDEO) (informational constructs) follow two separate routes to the rendering phase. The rendering phase does the final composition of the scene and projection to the two-dimensional (2D) space. This 2D coordinates space (RENDEREDVIEW) is mapped to the display coordinates in the display system. In the terminology of the graphics system [Pasman and Jansen, 2001], the interpreter and the compiler are referred to as the front end (frontEnd) and back end (backEnd), respectively. Bear in mind that the rendering of the scene graph is position and orientation dependent. The presentation filter [Oosterhoff and de Ridder, 2000] is an interesting component of the graphics system. Recall the alert message: "Battery low, immediate action required". If the graphics system decides to implement this as contrasting text, then the characteristics of the background on which the text is to be displayed has to be known and hence the camera construct. The property of the text is alerting. The precise attributes of the text (colour, size, face) are set by the presentation filter in a "late binding" fashion.

The geographic information system, (GIS), models real-world objects [Zlatanova, 2001]. The positioning system uses a skeleton view on real-world objects to carry out image-based position recovery. The skeleton (SKELETON) construct is dependent on the current position and orientation of the user for reasons of efficiency. Similarly, the GIS system provides the graphics system with a model of the real-world object that possibly obstructs a clear view of virtual objects: *occlusion*. The GIS system maintains a database of real-world objects. Two types of representations are used to address different aspects of real-world objects and their relations [Oosterom et al., 2002]: a geometrical and a topological representation. The *geometrical* representation is particularly suited for automatic incorporation of new data This representation stores separate objects as list of unique points. The *topological* representation is a representation with hierarchically structured objects. A house, for instance, is composed of a list of faces; faces can have sub-faces (doors and windows) etc. The topological representation is suited for generating virtual views of real-world objects. The informational objects SKELETON and OCCLUSION of Figure 2.7 are extracted from the topology database.

Figures 2.5 and 2.6 specify, to some extent, the structure of two selected components. It is important to realise that so far nothing has been said about their implementations; their engineering aspect. In particular, we anticipate that for certain applications, distribution (off-loading) of processes is required. The link between an encoder and decoder in Figure 2.5, for instance, will typically be implemented as a wireless transmission link. The encoder part of the source coder therefore can be situated at the backbone system and assigned ample resources. The decoder part will remain at the mobile terminal, which has only limited resources.

Figure 2.7 is a high level view on the structure that allows for transferring information back and forth a backbone system: the support flow (Section 2.2.1). The support flow uses a wireless channel as transmission medium. Like the source coding component, the components in the communication system have an encoder and decoder part for sending and receiving, respectively. The terms used for encoding and decoding are different though for each component. The encoding part of the channel coding

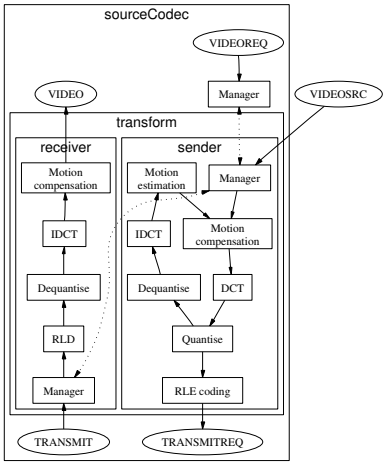


Figure 2.5: Source Coding component detailed.

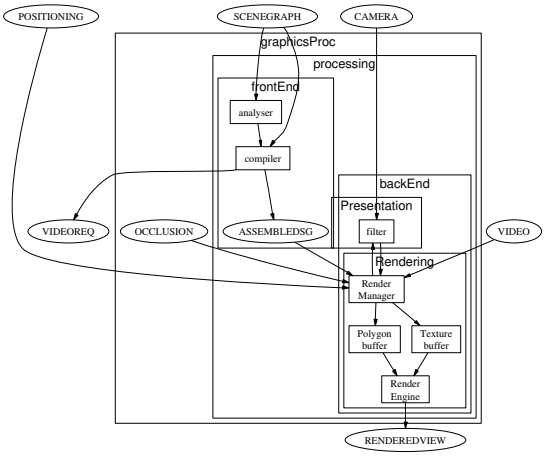


Figure 2.6: Graphics Processing component detailed.

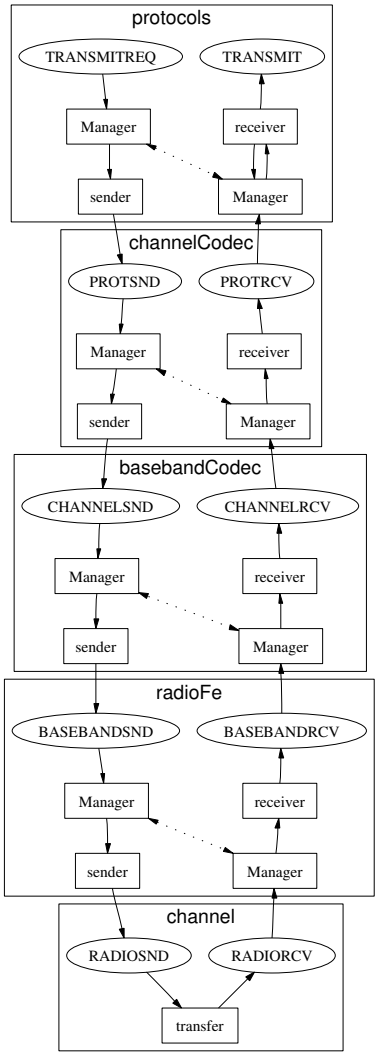


Figure 2.7: Ubicom communication system (support flow).

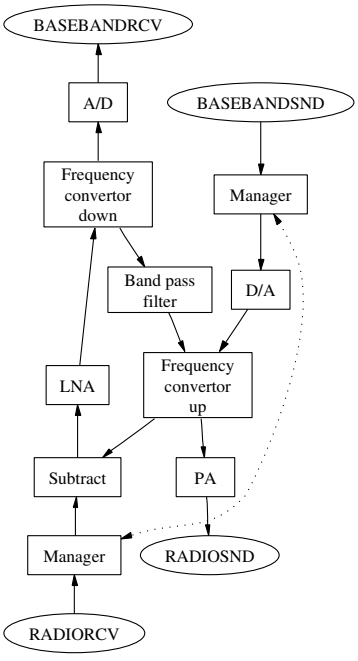


Figure 2.8: Ubicom analogue (adaptive) radio front end.

component, e.g., is referred to as (channel) encoding and decoding, while the baseband processing refers to its coders as modulation and demodulation with OFDM coding and decoding. The radio front end refers to encoding and decoding as carrier modulation and carrier demodulation.

Figure 2.8 illustrates in some more detail the computational view on the radio front end. The adaptive radio structure [Verhoeven and van den Bos, 2001] transfers a base

band signal over a wireless channel. Adaptivity is twofold: First, changes in the conditions of the wireless channel automatically induce calibration of filter parameters, i.e., the feedback control loop in the receiver part. Second, a tradeoff is possible between the reliability of data transmission and the power dissipation. This tradeoff is the result of combining flexible components. In [Tasic and Serdijn, 2002b], e.g., a method is introduced to visualise the tradeoffs possible when designing a voltage-controlled oscillator (VCO), which is a substantial part of the frequency converter (Figure 2.8).

Obviously the computational view details the processing of constructs from the informational view. As an example, in Figures 2.5 and 2.6, details of the underlying information processing structure are shown, which are hidden in Figure 2.4. The computational view also elaborates on the available option to map constructs from the computational view (processes) to constructs of the engineering view (processors).

2.2.4 Engineering view

Taking an engineering view on the Ubicom system reveals details about the implementation of computational constructs. The engineering view introduces the necessary *mechanisms* to assign functionality to processing components, i.e. *mapping*. Mapping of computational constructs to engineering constructs is a prerequisite for evaluating performance.

The integrity of the eventual system is determined by many aspects. The engineering view on the system maintains two important properties with respect to the fitness (suitability) and the perceptive quality of a Ubicom mobile terminal: energy consumption and form factor. Both properties are subject to coordination. One can regard the form factors being either the consequence of or the constraint to mapping and technology choices. Similar considerations apply to energy consumption and energy capacity.

It is instructive to start the discussion of the engineering view with some practical cases. During the course of the Ubicom project, three implementations were realised: 1) Strawman platform, 2) Paper platform (literature survey), and 3) Experimental platform.

The Strawman platform [van Dijk and van Reeuwijk, 1999] is functionally complete yet has limited mobility. The Strawman is a PC-based platform that uses a position recovery technology with confined coverage area, and offers mobility on a string.

A literature survey [Pouwelse et al., 1999] showed that at the time of writing of the survey it was indeed possible to compose a system that covers the basic functionality and has an acceptable form factor. Estimates of the energy consumption of the Paper platform were promising. The survey covers a limited (fixed) scenario and circumvents operational issues.

The Experimental platform is an heterogeneous platform with various interconnected processing cores. The basis of the platform is formed by the Lart; a dedicated low-power platform [Bakker et al., 2001] designed especially for this purpose.

Mapping of functionality links constructs of the computational view to constructs of the engineering view. All three platforms implement similar computational and informational graphs. However, different choices have been made, as each platform was designed with different concerns.

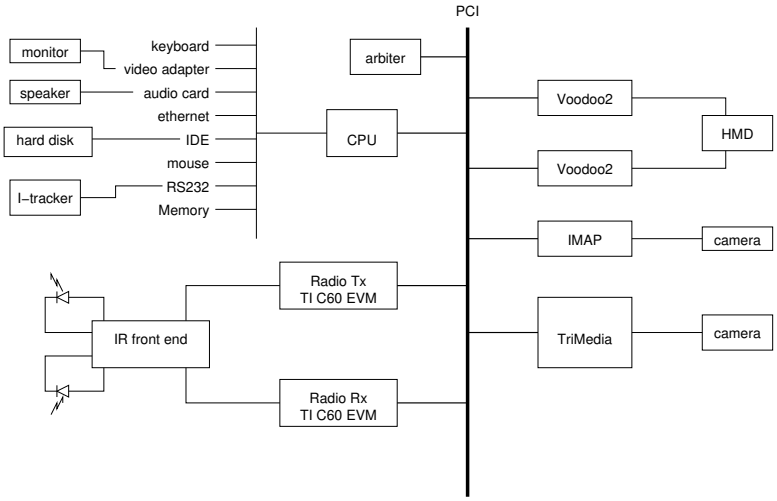


Figure 2.9: Strawman mobile terminal [van Dijk and van Reeuwijk, 1999].

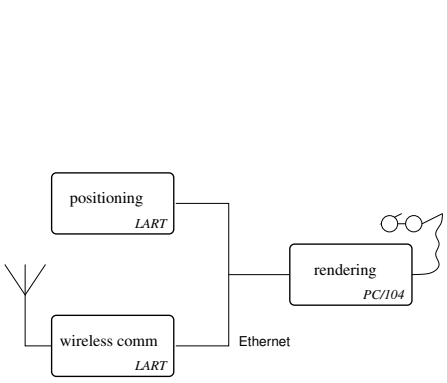


Figure 2.10: Experimental platform [Bakker et al., 2001].

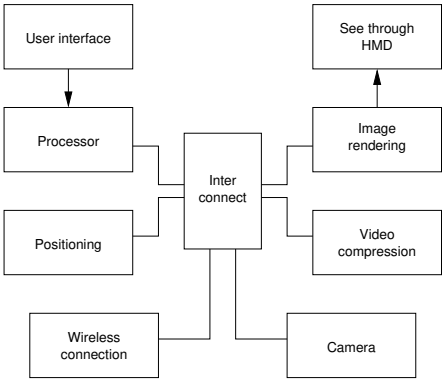


Figure 2.11: Literature survey platform [Pouwelse et al., 1999].

The Strawman mobile terminal is a single-bus architecture (Figure 2.9). The bus connects a high-performance general purpose CPU with dedicated peripherals such as a positioning system (l-tracker), rendering hardware (Voodoo2), source decoding accelerators (TriMedia), and baseband processing accelerators (TI C60). The Strawman implementation uses infrared communication. The mapping of computational constructs is straightforward. Constructs with dedicated hardware are mapped accordingly. All other constructs, except the video source and the corresponding encoding (Figure 2.5) are mapped to the mobile terminal CPU. The operating system (OS), which executes on the CPU, is an integral part of the engineerings view. The OS facilitates apparently par-

allel execution of multiple computational constructs on the same CPU core. We deliberately write "apparently"; the OS effectively implements mutual exclusion synchronisation between independent threads of control to intertwine execution of computation constructs. In the abstract, the process is observed as a true parallel implementation. However, particular choices of OS will influence performance and behaviour (information constructs). The Strawman uses a Linux kernel.

The Paper terminal that resulted from the literature survey takes a similar approach with respect to the mapping (Figure 2.11). The distribution of computational constructs is based on the graph of the computation view and an estimate of the limitations of applied engineering constructs (hardware components). In particular the on-terminal data transmission is considered a bottleneck.

The Experiment platform takes a more distributed approach. More functionality has been off-loaded from the terminal to a backbone system. The application manager as well as the graphics system front end is transferred to the backbone [Pasman and Jansen, 2001]. Also the texture sources and corresponding encoder are situated in the backbone, as is the case of the Strawman platform. Off-loading of computation- and storage-intensive tasks is necessary to arrive at a feasible mobile terminal layout (Figure 2.10), given external constraints such as a limited financial budget and available manpower. The position recovery system of the Experiment platform combines information from multiple sensors. The sensors reside at the terminal and the processing of their measurements is also done at the terminal. Envisioned, but not included in the current positioning system is image-based position recovery. Consequently the positioning system has a moderate performance and corresponding power dissipation. The Experiment platform runs Linux as its core operating system; the back end (rendering) part of the graphics system uses a real-time version of Linux. Computationally intensive tasks are assigned to dedicated processing units, which are connected with dedicated links. In Figure 2.10 Ethernet is used for these dedicated links. More suitable low power solutions are under construction; in particular a technique known as low-voltage differential signalling (LVDS) is considered.

The differences in choices for mapping and consequently the differences in engineering graphs has multiple sources. The differences can be ascribed to:

- Difference in attribution of properties.
- Evolution of technology.

Each platform attributes different values and priorities to possible performance properties. In general the design of any platform is a multiobjective optimisation problem. Assigning a weight (prioritise) to each of the performance properties reduces the multiobjective problem to a single-objective optimisation problem. Deriving the weight factors, explicitly or implicitly, is not a trivial task. An example of a process that derives these weight factors is the analytic hierarchy process (AHP) [Saaty, 1990]. Here, we took a pragmatic approach for each of the three presented platforms. The Strawman emphasises speed while neglecting accuracy, whereas the Experiment platform does the reverse. Mobility is essential for the Experiment platform, the Strawman only needed proof of concept. The engineering view introduces resource utilisation into the design. The mapping of constructs of the computational view to constructs of

engineering view thus effectively relates performance to resource utilisation: Quality of Service (QoS).

The evolution of technology (see also Section 2.2.5) opens up options that used to be beyond the budget. One aspect of advancing technology is covered by Moore's law (faster and smaller), another aspect is increase of functionality. The *mechanism* of how to enable evolution either by exercising mutationalism [Gould, 1977] or endosymbiosis [Margulis and Fester, 1991] is beyond the scope of this dissertation. What is important is the *policy* to incorporate and apply newly arrived technology.

In retrospect, the three platforms presented above show the exploitation of three types of advances:

i. Re-mapping of functionality (Algorithmic advances).

ii. Modification of efficiency curves (Technological advances).

iii. Extension of efficiency curves (Conceptual advances).

The essence of the system development problem is that implementations of computational constructs combine resources and functionality with a non-linear efficiency-capacity curve. As an illustration, consider Figure 2.12. From a development perspective, the ideal (or utopia) curve is a linear one. In that case algorithmic tradeoffs (mappings) can be freely made. However, in reality (typical a resource has only a small, usually non-linear, operation space. At best, a resource offers a number of operation points, e.g., through a composition of multiple modalities. Consider a wireless communication system that implements a range of commodity devices like short-range infra-red communication, medium-range Wi-Fi (IEEE 802.11B) communications, or long-range GSM communication. Technological advances extend the working area of a resource as well as they and improve the capacity and efficiency of resources (extended in Figure 2.12). A recent example is dynamic voltage scaling (DVS) of processing cores [Pouwelse et al., 2001]. Traditionally, a processing core offers two operation points: on (running) and off (idle). Technological evolution, Moore's law in particular, has increased the efficiency of the "running" operation point. People realised that processors are overprovisioned for most of their application areas, so a discrete set of operations points is now offered: idle, doze, sleep, etc. A proper combination of these operation points effectively lets the processor operate at quarter, half, or full speed, with corresponding energy savings. The obvious direction of evolution is to offer a continuous operation space. The dynamic voltage scaling paradigm offers a tradeoff between efficiency and capacity that may be exploited algorithmically. We comeback to this subject in Section 3.2.

The engineering view includes many more constructs than the two examples presented here. In Ubicom we introduced the concept of *generic buffers*. Generic buffers are flexible implementations of first-in-first-out (FIFO) uni-directional data paths. The idea of generic buffers is twofold. First, the computational view on the Ubicom system suggests a dominant data flow with occasional (control) events, see also Chapter 5. Second, generic buffers offer the flexibility to allow for simple remapping of constructs from a computational view to constructs of an engineering view, i.e., resources. Generic buffers are a significant part of so-called run-time systems. The run-time system provides

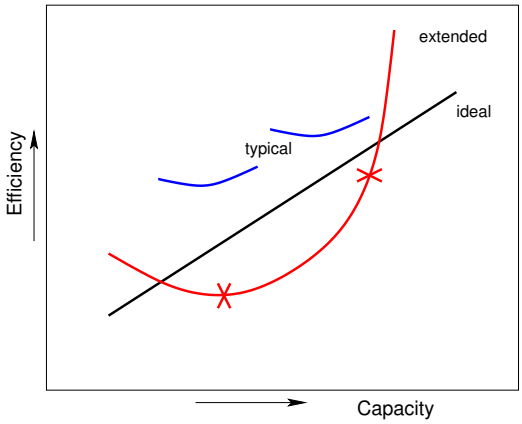


Figure 2.12: Efficiency Curve (trend)

means to execute a parallel program, particularly on heterogeneous hardware. Flexible run-time systems are extensively used for automatic mapping of computational constructs. SPAR [van Reeuwijk et al., 2001] is an example of such an automatic mapping and execution tool. SPAR is a Java extension that specifies computational constructs (a sequential program), which are mapped onto constructs of an engineering view. The run-time system is the "glue" for parallel execution.

We stressed the relation between the computation view and the engineering view as a mapping of computation constructs to engineering constructs. However there is also an action in the opposite direction. Engineering constructs (or resources) have limitations. In a context-aware system, the capacity of resources influences the behaviour of computational constructs. The necessary information about available resource capacity is communicated through informational constructs. The informational view that results from the link between engineering and computational constructs is generally known as *performance modelling*. As an example, van Gemund has developed an analytical performance modelling methodology [van Gemund, 1996]. The method takes a computation model, an engineering model, and their mapping and derives an analytical performance model. The strength of an analytical model obviously is its potential to swiftly explore a broad design space and to perform optimisations. Other researchers developed methodologies based on extensive simulations. The methodology of Kienhuis automatically generates executable simulation models based on a computation model, an engineering model, and their mapping. Because of the automatic generation, a method known as design space exploration is possible through repeated simulation runs [Kienhuis, 1999]. We will return to this matter in Section 3.1, in particular in Example 3.1 on Page 38.

2.2.5 Technical view

A technical view addresses the technological aspects of a Ubicom system. Technological aspects are dominated by practical issues, like the availability of standards, software, and hardware, but also limitations to, for example, time and manpower budgets play an important role. Flexible designs require flexible technology. Research projects like the Ubicom project therefore rely on open standards, open source, and free[1] technology: hardware and software. Closed (or proprietary) systems can be applied, but usually at the cost of wrapping and consequent inefficiencies. Through wrapping – encapsulation of the closed standard (endosymbiosis) – one can achieve the necessary flexibility; this proved a suitable approach, albeit a suboptimal one.

Technological aspects were briefly considered in the views discussed above. With respect to standards, the internal video format (Figure 2.5 on Page 27) is a DCT-based compression format, e.g., ITU-T H.263, ISO MPEG-2/4. The SCENEGRAPH informational object (Figure 2.4 on Page 24) is an ISO VRML 97 dialect. A system that complies to standards offers the advantage of reuse of components developed elsewhere, and the disadvantage of limited flexibility. In Ubicom the applied VRML dialect extends the standard VRML language with dynamic constructs.

The support flow (Figure 2.7) uses OFDM coding and QAM modulation for each (orthogonal) carrier. This technique has been chosen because it offers high throughput and requires little (frequency) bandwidth. OFDM coding effectively mitigates sincere channel fluctuations caused by multi-path interference. The choice for this type of modulation imposes additional constraints on the behaviour of the radio front end (radioFe). In particular, linearity of amplifiers [van den Bos, 1999] is crucial for correct channel demodulation and decoding.

It is not necessary to give an exhaustive list of all standards and techniques applied in the Ubicom system here, but a few more examples are appropriate though. The consulting radio front end [Verhoeven and van den Bos, 2001], for instance, applies an approach of combining multiple designs. At run-time, the system adapts to varying conditions and powers up only the appropriate design. This results in a slightly larger silicon area, but with improved performance and decreased power dissipation. Likewise, channel coding combines convolutionary and block coders at the encoding side with so-called turbo decoding techniques at the decoder side. Again, on the basis of design-time and run-time considerations the appropriate parameter settings are chose to arrive at efficient implementations. As a final example we mention database organisation of the geographic information system. Designing and implementing a database system from scratch is in most cases not feasible. Commercial, and thus proprietary, solutions are available, however. The implementation of the aforementioned geometrical and topological organisation of geographical data is based on a commercial database system (Oracle database server). Notwithstanding the wrapping penalty, the performance of queries is satisfactory.

[1] Free as in “free speech; not free beer” [Foundation, 1984].

2.3 Discussion

The preceding sections introduced an ontological inspired framework that relates different views on the same (beyond our reach) architecture of a system. We used the Ubicom system to exemplify a set of related views. A recurring concern is the interdependency among constructs part of the same view and among constructs from different views. To develop a proper system, one needs to achieve clear coordination among subsystems. Without proper coordination a system degrades to a mere aggregate. Proper coordination ensures the integrity of the system and prevents a functional breakdown. We identified three important aspects for organising coordination properly:

- Abstraction
- Adaptation
- Cooperation

Regarding a system in the abstract, limits the set of predicated properties. Abstractions were first introduced in the informational view. But in a sense every view is an abstraction. The ability of a system to adapt to its immediate environment implies that the system values the properties it has in common with the environment. Constructs of the computational view apply adaptation directly. A system reacts to a change in the environmental conditions. An indirect example of adaptation is found in our discussion of the engineering view. We discussed the three platforms and saw that the performance of engineering constructs caused a change of behaviour in computational constructs. The selection of an engineering construct and the consequent adaptation of a computational construct is reflected in an informational construct. Equipping systems with an awareness of changes in its properties can contribute positively to the system integrity: cooperation. Cooperation implies that a system is aware of the influence that changes in properties have on its environment. In a way, cooperation is the return path of adaptation. Adaptation is an inbound bonding relation from the milieu to the system (which implies action) and cooperation is an outbound bonding relation from the system to its milieu (reaction). So, together adaption and coordination create interaction.

The presented views do incorporate collaboration to some extent, but only in an implicit way. The developed viewpoints are the product of an authoritative and centralised design method. In the case of the described system in this chapter we maintained a database of dependent views from which the individual views are extracted. A viewpoint defines the concerns that select entities and relations from the database. These entities and relations form consequently a view. The viewpoints are perfectly suited for analysing and reasoning about the system. In fact this is a requirement in order to guarantee the consistency of the database. Flexibility, however, is all in the mind of a single entity: the system architecture team. In the following chapter, views are developed collaboratively by the respective developers of individual subsystems.

Three

Negotiated Quality of Service

QUALITY of service (QoS) is an overloaded term. Definitions of QoS greatly depend on what view is chosen on this subject. We restrict our treatment to two basic views on QoS: an informational view and a computational view. An informational view on QoS attributes predicates to selected properties of constructs from an engineering view on the system, which makes non-functional aspects explicit. Typical examples are the *capacity* of a channel and the *distortion* of signals (signal-to-noise ratio, SNR). A computational view on QoS concerns the protocols for exchanging and handling predicates from the informational view. Both views are significant for our observation that a QoS framework implements the coordination among components in a communicating system.

In this chapter we derive the relevant aspects of such a QoS framework. Subsequently we present a compliant framework: adaptive resource contracts (ARC). In Chapter 2, we concluded that any coordinating framework must support the following three aspects: abstraction, adaptation, and cooperation. The reason for developing yet another QoS framework is that existing frameworks do not satisfactorily cover the cooperation aspect. Even though modern QoS protocols [Aurrecoechea et al., 1999] facilitate negotiations, their information exchange is unidirectional and, what is more, the information usually does not include the non-functional aspects of *resource utilisation*. The flow of information of existing protocols for QoS either communicates predicates from constructs of the engineering view of a system to constructs of the computational view of a system or the other way round, but never both ways. This unidirectional sharing of information would support adaptation, however; bidirectional sharing of information is required to properly support cooperation. Effective cooperation requires that components are truly context aware. Another reason for developing ARC is that resource utilisation metrics have only recently been included in the informational constructs of QoS frameworks [Yuan and Nahrstedt, 2001].

In Section 3.3 we will use the developed views on QoS to compare the development choices made in ARC with related work. In part the comparison is made using an conceptual QoS framework. We conclude with a discussion on the applicability of

ARC to coordinate components in a complex system. ARC trades off costs and benefits, which is a multiobjective optimisation problem that any QoS framework must solve. Optimisation is the topic of Chapter 4.

3.1 Views on QoS

Our approach to quality of service (QoS) follows our observation that a QoS framework implements the coordination among the components of the communicating system. Two aspects of a QoS framework are important, **what** qualitative information is handled by the QoS system and **how** is this information being exchanged. The former is elucidated in an informational view on QoS and the latter is elucidated by a computational view on QoS.

In Section 3.1.1 we argue that non-functional aspects must be shared among components and in Section 3.1.2 we argue that the effective sharing must be implemented distributively. As a consequence we conclude that cooperative components in a QoS framework are necessarily context-aware. A component must be aware of the fact that components in its immediate environment may change their state. Equally important, a component must be aware of the fact that a change of its own state may affect the state of other components.

3.1.1 Informational view on QoS

In Chapter 1 we have discussed the issue of cooperation and the eternal struggle for balance; performance, behaviour, and resource utilisation. Making a proper tradeoff implies making a well-informed tradeoff. The following example indicates what information is required for finding the balance.

Example 3.1 (Scaling) *Recent – software-only – video decoders support mechanisms to maintain real-time decoding while sacrificing quality. Low-latency decoding, which suggests real-time behaviour, can be guaranteed continuously at the expense of increased distortion (through skipping video frames). Given a high-performance platform, with ample access to* CPU *and memory resources, it is quite possible for a video decoder to display all frames of a selected video stream. Two possible sources causing disturbance of the decoding process are apparent: access to resources and availability of resources. If for some reason the availability of* CPU *or memory resources is insufficient, then actions have to be taken in order not to delay the decoding process. In this example we assume active skipping of frames. Access to resources, in particular access to memory, is predominantly determined by the organisation of the encoded stream. Complex encoded streams have complex interdependencies and therefore require a high memory bandwidth [Patterson and Hennessy, 1990]. Selective skipping of frames reduces the workload on the memory bandwidth. To reduce the utilised memory size, a different selection of frames to be skipped is required. Both selection schemes result in real-time decoding.*

Implementation matters. The quality of a displayed video does not only depend on how it is perceived by the client, but also on the capabilities of the platform on which the decoding process is executed. An informational view on QoS therefore must include aspects from computational and engineering views on, in case of the example,

the video decoding system. Informational QoS constructs thus require at least metrics for *distortion* and *capacity*. In the example, a client to the video system requires information about the signal-to-noise ratio (SNR) of the video sequence after transmission and decoding. The video decoder, being a client of the processing platform, is interested to have information about the available CPU and memory capacity.

Still, there is additional information that must be shared. The following example illustrates that sharing of information about resource utilisation is of particular interest among the subsystems of a communicating system.

Example 3.2 (Sharing) *Assume an application that combines a day view and a night view on a scene. The application uses two concurrent software-only video decoders from* Example 3.1. *The first decoder decodes the day-view stream and the other decodes the night-view stream. Note that the characteristics of the streams are hugely different. Assume an application that is best suited with both views presenting a comparable display quality. When the resources of the platform are limited, the individual decoders have to sacrifice quality. The application coordinates the tradeoffs; either explicitly or implicitly. Basically the application assigns a budget of available resources to the individual video decoders.*

The above example illustrates once more that implementation matters. When resources are shared, information about the resource utilisation of individual components must be conveyed. Neglecting this information will jeopardise the integrity of the system.

In retrospect, looking at the informational QoS constructs we deduce that a computational view on the system contributes a metric for *distortion*, an engineering view on the system contributes a metric for *capacity*, and the mapping of computation to engineering constructs contributes *resource utilisation*. Recall that the creation of the informational QoS construct was inspired by the design constraint to support evolution and to organise coordination in a distributive and non-iterative way (see problem description on Page 4). In effect, the informational QoS construct abstracts the behaviour and the performance of individual subsystems and instruments their adaptation and cooperation.

3.1.2 Computational view on QoS

Example 3.2 considered a straightforward system: one application, two concurrent video decoders, and a single CPU. Already in this clear arrangement, exchange of information is required for proper coordination. There are many ways to organise coordination. A practical procedure, however, must be1) non-iterative and 2) distributive.

The non-iterative requirement has two grounds. First, for a run-time implementation with real-time constraints, an iterative procedure is less attractive. Second, in a communicating system in which subsystems possibly stem from very different domains of expertise, iterative methods will fail to result in prompt delivery of designs. This is caused by the possible long turnaround times of the design methodologies that are applied in the respective domains.

The distributive requirement is a direct result of the complexity of the system. Developing a central coordinating entity would require a single framework that models all

subsystems concurrently. In the preceding chapter we already argued against describing a system in a single stroke. Another argument in favour of a distributive procedure is the fact that complex systems are in constant state of *flux*. A possible central authority would be required to continuously adapt the framework in which the coordination is evaluated. Or hypothetically, it would require a metaframework that encompasses all possible scenarios of system composition. In these cases, a distributive coordination framework is more practical, since it has necessary flexibility to adapt itself. The underlying metaframework presumably exists, but will never be manifest. Obviously bidirectional and cooperative sharing of information among the components of the system is required. The applied QoS protocol thus must support bidirectional (cooperative) information exchange. QoS negotiations protocols are typically applied to coordinate systems that are in a constant state of flux.

3.2 Adaptive Resource Contracts (ARC)

A QoS framework that complies with the preceding contemplation of a proper framework for implementing the various aspects of QoS is Adaptive Resource Contracts (ARC). The framework presumes a system with democratically interacting subsystems, an heterarchy [Dilts et al., 1991]. Sharing of information among distributed subsystems (peers) are typical client-server interactions. One subsystem offers services (the server), which are applied by another subsystem (the client). However, it is not at all obvious whether the offered or the requested service is authoritative, even if it were obvious then the situation may change during operation. In addition, subsystems are part of a larger composition, so they act as a client as well as a server and consequently will have has a client-side as well as a server-side interface. Thus subsystems are individually responsible to maintain the necessary domain knowledge but they have joined responsibility for the integrity of the system as a whole.

The ARC framework eventually establishes contracts between client-server pairs. A contract combines two basic kinds of contracts: resource contracts and adaptive contracts. Resource contracts implement the requirement of going beyond an abstract, functional-only, QoS interface between client and server. ARC explicitly incorporates resource utilisation metrics in its interface parameters. Adaptive contracts implement the coordination protocol among clients and servers. A client and server establish a bipartite contract through a three-sweep protocol of request–offer–select. If necessary, either of the two parties may initiate renegotiations.

The ARC framework has the required transformational properties. Because every subsystem plays twin rôles – being a client as well as a server – it effectively translates and interprets the context offered by its employed services into the context of its clients. The translation and interpretation process includes the contribution of the subsystem to the system properties as a whole. Thus at the cost of showing context awareness, subsystems can concentrate on domain specific issues and still contribute to the integrity of the system; a significant step in mastering the complexity of communicating systems.

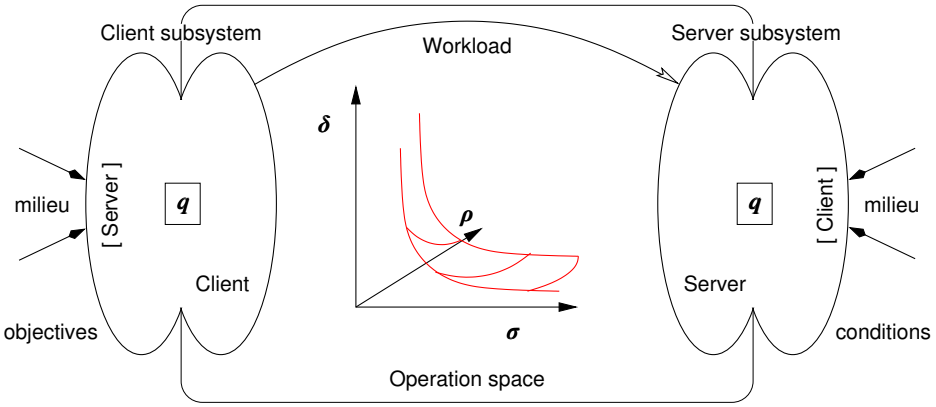


Figure 3.1: ARC informational view.

3.2.1 Resource Contracts

In a communicating system, communication amongst subsystems is between *clients* and *servers*. The client takes services from the server while inducing a *workload* on that server. The shared milieu of client and server is captured (attributed) by their mutual so-called *operation space*. The composition of the client and server systems yields a system again with a client interface and a server interface. At the client interface the newly composed system requests services, i.e., the conditions under which to operate. Whereas at the server interface, the newly composed system offers services, i.e., the objectives with which the system must comply. Figure 3.1 shows a diagram of a compound system: a client, a server, and their operation space. These constructs constitute the informational view of ARC.

The operation space of the ARC framework attributes the mutual milieu of the client and server under observation. The operation space of ARC is a tuple with three classes of attributes (parameters), each class being a vector: distortion ($\boldsymbol{\delta}$), capacity ($\boldsymbol{\sigma}$), and resource utilisation ($\boldsymbol{\rho}$). The division of parameters in three distinct classes has a practical cause. Observations during system development of collaborations amongst subsystems in different application areas showed that the aforementioned parameters are often recurring identifiers. ARC, however is not strict in imposing this set of parameters, but for the sake of the discussion we refer to the tuple $\langle \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\rho} \rangle$ as parameters spanning the operation space.

An operation space is a parameter space with operation points, which specify an optional contract between client and server. An operation point corresponds to a mode of operation of the combined client and server subsystem. From the perspective of the client, an operation point is a model of the service in terms of its performance and behaviour: it shows how good the service is, how much of it is supplied, and at what cost. From the perspective of the server, the operation point is a model of the workload.

An operation space typically consists of more than one option. There exist more

ways in which a client and a server can cooperate, since both support a range of scenarios. As a result the emerging properties of the compound system (combined client and server system) are affected by the selected operation point. The emerging properties themselves contribute to the operation space of the compound system, and as such, model the quality and resource utilisation of the compound system. The attribution of these emerging properties is context dependent; the role the compound system has in the total system determines the manifold from which its properties are valued. As an example, some types of applications benefit from a video communication system that supports a progressive range of video quality sequences, whereas other types of applications adhere to an all-or-nothing strategy. Now consider a particular video communication system that offers an operation point at its server side (conditions) with moderate quality at low resource utilisation. The first type of application will value this specific system as suitable whereas the second type of application will value this specific system as useless.

To state the foregoing more formally: when offering an operation point, a flexible system component must be aware of the varying objectives of its clients. In general, generating a single operation point is inadequate, since a single operation point would only be sufficient in the rare case that the conditions offered by the server exectly match the objectives imposed by the client. A single proper operation point thus implies detailed knowledge about the internals of the client and the context of the client. Instead, a flexible component may generate a whole set of operation points, each point not dominating the others. The set circumvents the need for in-depth knowledge of its clients. The set of operation points is the solution to a so-called *multiobjective* optimisation problem [Miettinen, 1999]. A set of non-inferior (or non-dominant) points is usually referred to as a set of Pareto points [Pareto, 1896]. Note that it takes expert knowledge to generate such a set. At first sight, having a set of points rather than a single point in the operation space may seem an example of overprovisioning. However, a set broadens the scope of the client and therefore reduces the number of communication iterations; it is a prerequisite for non-iterative implementations (Postulate 1.2 on Page 8).

The operation space is the result of the collective of a client and a server: both contribute parts to the operation space. Their individual contributions are the result of an internal evaluation process. The client as well as the server collects (decision) variables in a vector $\boldsymbol{q}$, see also Figure 3.1. Different settings of these variables (vector entries) transform to different entries in the operation space. A straightforward optimisation that makes communications more efficient is to offer only those operation points that are Pareto points, so operation points that are optimal in at least one dimension of the operation space. A more mathematical discussion on ARC will be presented in Chapter 4.

The immediate environment (milieu) of a system component affects the precise value of operation points. The milieu of a system component includes its *objectives* and *conditions* as set by the larger system. In the diagram of Figure 3.1 the objectives of the compound system are part of the milieu of the client subsystem. Similarly, the conditions for the compound system are part of the milieu of the server subsystem. Internally, the client and server have an overlapping milieu; they share an operation space. The system of Figure 3.1 effectively translates (transforms and conveys) its

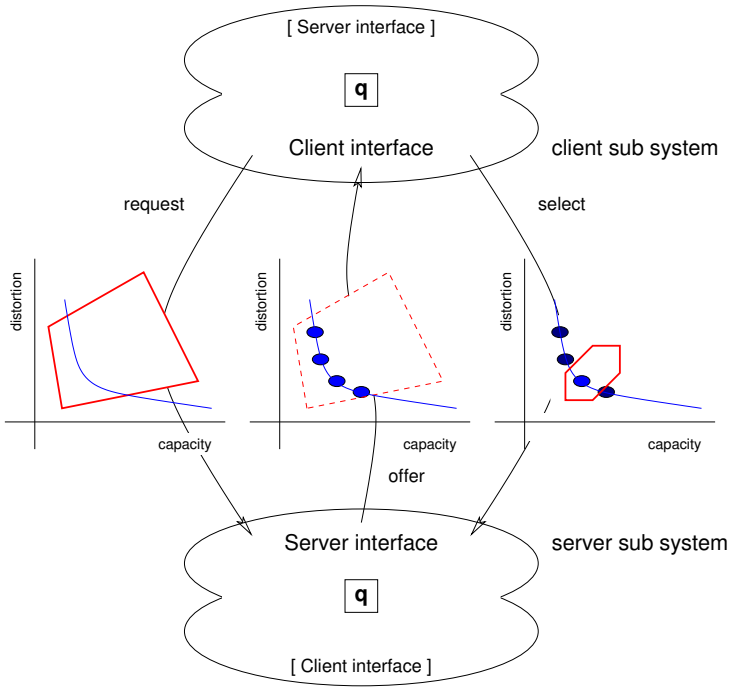


Figure 3.2: ARC three-sweep protocol.

objectives and its conditions.

The operation space is the result of the cooperation between a client and a server; it is an explicit account of the context awareness of system components. Efficiency reasons often contribute to the eventual appearance (scope and dimension) of the operation space. For instance, in [Lieverse et al., 1999], high-level design space explorations are applied to deliberately limit the scope of operation space parameters. Another example is making the interdependence of operation space parameters explicit. At design time the relative precedence of parameters can be set [Saaty, 1990], which makes run-time implementation more efficient. By exploiting the static precedence among parameters, one can effectively limit the dimensions of the operating space; an example of this can be found in Section 6.1.3.

3.2.2 Adaptive Contracts

In this section we present a protocol that implements adaptive contracts. ARC applies a three-sweep protocol: request–offer–select, which is outlined in Figure 3.2. In the first sweep the client subsystem issues a *request* to the server subsystem. The server responds – second sweep – with an *offer*. The client subsequently *selects* in the third sweep an appropriate operation point, which finalises the terms of a contract.

A request, typically, contains a range of objectives. A request defines an (open) interval in the operation space. Given the objectives and given the conditions, a server subsystem can evaluate the consequences of setting decision variables ($\boldsymbol{q}$ of Figure 3.2). Evaluating the decision variables for a range of settings yields a set of Pareto operation points: the offer. The contract, finally, specifies a small interval round a selected operation point from the offer. The interval leaves room for both the client and server to perform local adaptations without negotiating a new contract. The example below illustrates the ARC protocol in a heterarchic structure.

Example 3.3 (Video communication) *Consider a mobile video communication system [Taal et al., 2002]. A mobile terminal is equipped with a camera. The video sequence taken from this camera is encoded, transmitted to a base station, and decoded again; all in real time. Assume a straightforward engineered mobile terminal: a battery-powered terminal with a single CPU running a multi-threaded operating system. For the sake of the example, the operation spaces between the various computation constructs are simple.*

Figure 3.3 *combines the informational, computational, and engineering views on the system. At the top of the diagram, the user requests an interval of distortions. The manager translates the objectives of the user into an interval of distortions and video resolutions. The source codec evaluates the source characteristics, takes into account the objectives, and evaluates internal decision variables. This results in a specification of an interval of throughput and bit error rate (*BER*) objectives. The specification is put as a request to a channel codec. The channel codec uses a similar approach to translate (cast) incoming objectives into a request for data transmission to the transmitter.*

Finally, the radio front-end (transmitter) senses the wireless channel conditions. These conditions together with the request from the channel codec make up the context in which the radio front-end solves its multiobjective design problem. The result is an offer to the channel coding layer. The offer from the radio front-end to the channel codec completes the context of the channel codec and therefore provides sufficient conditions to solve its multiobjective design problem. Offers ripple up the heterarchy.

Eventually the user receives an offer with multiple fully-determined operation points. The user selects an appropriate one and establishes a contract with the manager. The manager knows the settings of its internal decision variables that correspond to the selected operation point. The manager also knows the correspondence to the operation point of the offer it previously received from the video encoder. Contracts and settlement of internal decision variables ripple down the heterarchy until finally the radio front-end receives the contract information; processing can begin.

The ARC three-sweep protocol compares to an intuitive approach for exchanging context information, which can be observed during the exploratory stage of system development, especially when this development takes place in a multidisciplinary setting. During the exploratory stage there is regular interaction among component owners; individual owners try and establish the context of the component they are responsible for. During these interactions, vague requests are made, offers are expected in return, and sensible operation areas are selected.

For reasons of efficiency, run-time implementations usually limit their scope and flexibility. The ARC framework is suitable for determining where flexibility can be sacrificed. One option is to exploit the possibility that similar evaluations take place during all stages of the development. For example, during system analysis, the system

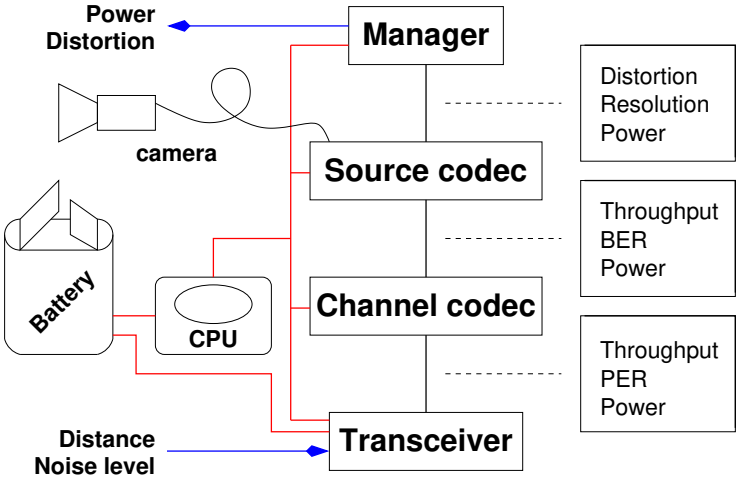


Figure 3.3: Video communication system.

is submitted a scenario. Subsequent evaluation may show that a component uses only a limited number of its potential operation modes. It therefore is sufficient to support only those modes of operation that are actually used at run-time.

Possible changes in system objectives or system conditions will propagate the heterarchic structure of the system. The reach of the changes may differ, though. Some changes affect the entire system, whereas others vanish while traversing the structure. Less significant context variations can be resolved locally, exploiting the ability of a component to adapt to a changing milieu. More significant context variations require the collaboration of more than one component; in that case renegotiations must be started.

3.3 Related work

In this section we give an overview of the vast amount of literature on the subject of negotiated QoS. In the ARC framework we advocate cooperation as the means to capture the dynamics of a system so as to coordinate resources and services among system components. This specific topic and its relation to existing literature is discussed in Section 3.5.1. We do not intend to be exhaustive in referring to related work, yet relevant concepts are put into perspective. References in this section might prove useful for run-time implementation of the concepts laid out in this chapter.

Quality of service has been addressed by a large number of researchers, general surveys of QoS architectures and QoS concepts are presented in [Aurrecoechea et al., 1999; Nahrstedt et al., 2001]. A common technological argument against implementing QoS architectures is overprovisioning; when implementing an abundance of resources, sharing of resources is never necessary. Nahrstedt gives in [Nahrstedt, 1999] three arguments against overprovision: greedy applications might block a resource entirely,

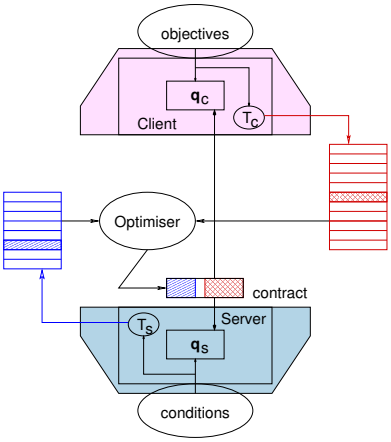


Figure 3.4: Generic negotiated QoS constructs.

head-of-line applications in a FIFO queue may block time sensitive applications down the queue, and last-mile problems hinder end-to-end QoS guarantees.

In Section 3.1 we developed two views on QoS, an informational view and a computational view. The relevant issues addressed in these views are visualised in the diagram of Figure 3.4. The diagram shows a client and a server. Both the client and the server export multiple multi-modal parameters at their QoS interface. The predication of these parameters is done in the context of the respective components; the objectives of the client and the conditions of the server. The interface parameters, in turn, are input to an optimisation process that yields a bilateral contract. The contract is an explicit account of a reservation. In the generic case, the optimiser continuously monitors the current state of the exposed interface parameters; feed forward from the client and feedback from the server.

In Section 3.3.1 we regard the related work from an informational perspective, concentrating on the operation spaces and generic coordination. In Section 3.3.2 we take a computational perspective and review a number of systems for negotiated QoS.

3.3.1 Informational view on QoS

Adaptive schemes

Figure 3.4 represents a *reservation*-based QoS scheme with an explicitly established contract, whereas schemes with implicit contracts are referred to as *adaptive* schemes. Typically, adaptive schemes do not have a separate optimiser and lack communication of context information in a proper format. Instead, adaptive schemes use extensive processing to derive the context of the underlying optimisation problem and then effectively adapt to changes in their immediate environment. Steenkiste identifies two distinct "adaptation models" (performance-based and model-based ones) and a hybrid one (a feature-based model) [Steenkiste, 1999]. The performance-based ad-

aption model corresponds to a traditional control loop, where the adaptable process monitors the performance of employed services. This type of modelling corresponds to the adaptive schemes mentioned before. Secondly, Steenkiste recognises a model-based adaptation model, where the adaptive process models its interior process based on external service performance parameters. We successfully applied the model-based adaptation model in a video-over-wireless experiment [Taal et al., 2002].

Adaptive schemes for QoS assurance that are based on feedback control are suited for analysis. In literature a number of control theoretical approaches have been reported. Abdelzaher et al. for instance, equipped a web-server with a classical (deterministic) proportional integral (PI) controller [Abdelzaher et al., 2002]. Another example of feed-back control is found in [Li and Nahrstedt, 1998], which uses a classical proportional integral derivative PID controller and more recently they include prediction by means of a Kalman filter [Li et al., 1999]; see e.g. [Kailath et al., 2000] for an introduction of Kalman filters. The experiments in [Li et al., 1999] demonstrate that an (artificial) tracking system breaks down without adaptation, whereas with adaptation the system maintains its specified tracking precision even under harsh conditions. In [McNamee et al., 2000], the QUASAR QoS framework is described, which considers streaming applications: a cascade of pipelined processes. The choice of controller is again a PID controller, which is a generic part of their encapsulated feedback framework: SWIFT. In [Steere et al., 2000] the SWIFT framework is applied in an experiment with a simple two-stage cascade. In [Koliver et al., 2002] feedback control loop is implemented with a so-called fuzzy controller. The advantage of fuzzy control over deterministic control, e.g., implementing a PID controller, obviously is its ability to dynamically adapt the behaviour of the controller. A disadvantage is the lack of a theoretical framework to reason about the performance of the system.

Chen et al. compare an adaptive scheme with a reservation- based QoS scheme [Chen et al., 2001]. It turns put that either scheme can be used in their situation; the adaptive scheme requires little initialisation work and much work during operation, whereas the reservation-based scheme does most of the work during initialisation and operation becomes simple. The researchers eventually favour the adaptive scheme. However they pass over the issue of evolution and exceeding domain knowledge.

Format and exchange

QoS assurance requires effective communication of information. In the ARC framework the specification of this information is left open. The specification is very application dependent, a universal form for all specifications in a complex system would therefore be impractical. Bhatti and Knight take a similar approach in [Bhatti and Knight, 1999]. They communicate context information through so-called QOSSPACES, which form overlapping compatibility requests.

Other researchers implemented different quality-description and quality-modelling languages, QDL and QML respectively. In [Loyall et al., 1998] a plethora of QDL dialects is developed for describing contracts, structure, and resources. Each QDL dialect has been designed with a specific application area in mind: multi media streaming applications. Also their examples are oriented towards structures with two layers: the application layer and the network layer. This makes them less suitable for every QoS

interface in a complex system. QML has been developed in [Frølund and Koistinen, 1998], which is similar to QDL but apparently more flexible. QML has three main constructs: contract type, contract, and profile. A contract type defines the QoS category (reliability or performance) and its dimension (set or enumerated). A contract is an instantiation of a contract type. A profile associates contracts with interface entities. As a final example of a quality description language is a QoS model based on the formal specification language Z [Staehli et al., 1995]. Staehli et al. defines three concepts: content, view, and quality. Content is an characterisation of the source, view an ideal presentation of the content and quality is a measure of the distortion of the eventual presentation. The QUASAR QoS model of [Walpole et al., 1999] refines the QoS model of Staehli et al..

QoS assurance through QoS adaptation requires a measure of optimality. Most frameworks implement this through *value* functions. Value functions map QoS related parameters to normalised predicates, they appear in various forms in literature. To give some examples. Chatterjee et al. use in their QUASAR project benefit functions to predicate the benefit of a selected mode of operation Chatterjee et al.. The QUASAR QoS model is another example that uses *utility* functions, a normalised measure of usefulness [Walpole et al., 1999]. As a final example the Q-RAM project applies utility functions that combine a resource consumption surfaces with a utility surface [Rajkumar et al., 1997]. Both surfaces are a function of one or more QoS parameters. The resulting utility function effectively trades off resource usage and (user) utilisation.

3.3.2 Computational view on QoS

QoS architectures, in particular their run-time implementations, are presented as middleware [Agha, 2002; Tripathi, 2002]. The underlying idea is to clearly *separate* concerns of functionality and performance optimisations; a form of aspect-oriented programming [Elrad et al., 2001; AOSD-web, 2001–2002]. The QoS functionality is implemented in what Schmidt refers to as "common middleware services" [Schmidt, 2002]. Basically it is a set of programming interfaces for QoS and resource coordination that is common to a specific range of application areas. Because middleware services have a strong orientation on run-time implementations, they lack the flexibility that is required in complex systems. Existing middleware implementations therefore target specific, limited, application areas. In [Tripathi, 2002], the research community is challenged to develop true "policy driven middleware for computer supported cooperative work"; ARC might very well prove to be a good start.

Reflective middleware [Kon et al., 2002] is a generalisation of middleware. This class of middleware has the ability to reflect, that is, to access, reason about, and alter its own interpretation. Reflection requires a metasystem as implied by Gödel's incompleteness theorem[1] [Hofstadter, 1985]. We already rejected this approach for ARC.

As mentioned before, there are many schemes for negotiated QoS; some are heuristical whereas others are more organised. In Figure 3.4 we presented a generic framework for negotiated QoS with distinct components for distinct concepts. Practical frameworks for negotiated QoS usually combine and confine concepts as not every

[1] "If the formal system P is *consistent*, its *consistency* is *unprovable* within P" [Braithwaite, 1962].

concept maps to a distinct component and not every concept is implemented in all its possible detail. The ARC framework, e.g., combines the optimiser and client-side operation space with the client-side evaluation. Figure 3.5(a) is a conceptual diagram of the ARC framework drawn in the generic negotiated QoS framework of Figure 3.4. In this section we review a number of frameworks for negotiated QoS.

The most straightforward negotiation scheme is *call admission*. At the time of establishing a service employment, the client specifies its requirements; the server tries to allocate the necessary resources on the route to the final service and reports success or failure. Call admission is visualised in Figure 3.5(b). The client provides a single request. The underlying server simply matches its current state with the request and acknowledges or rejects the request. Call admission does not support negotiations during a call, so applications tend to be greedy and allocate enough throughput to accommodate their most demanding mode of operation. Usually this worst-case allocation behaviour locks up network resources that could be used by others.

Value functions are applied in multiple frameworks. Value functions propagate information *down* the structure, which is generally not a good idea. Information accumulates in the bottom layers of the system, and consequently decisions have to be taken at a distinct location. Moreover, the time scale in the bottom layers is much smaller than the scale at layers higher up the structure. Examples of systems that apply value functions are QUASAR [Chatterjee et al., 1997], ERDoS [Lee and Sabata, 1999], and Q-RAM [Hansen et al., 2001; Lee et al., 1999b; Rajkumar et al., 1997]. In [Bianchi et al., 1998], value functions are applied for utility-fair scheduling. The corresponding QoS structure uses local server-specific adaptations. See the diagram of Figure 3.5(c) for a conceptual view; systems that are based on value functions, typically implement some sort of centralised control.

The classical control loop applies feedback control, thus effectively propagating information *up* the structure. Given the argument on the grain size of time above, this is a potentially better approach than feedforward. However, because of the difference in time granularity between the top levels and bottom levels, controlling the lower components from a distinct top-level component remains awkward. Feedback control can be useful though in situations where services can be characterised with one parameter and have relatively small fluctuations. Both the AQUA [Lakshman and Yavatkar, 1996] project and the SWIFT [Goel et al., 1999] project implement generic support for feedback control. See Figure 3.5(d) for a diagram.

Quality events [West and Schwan, 2001] is a framework with monitors and handlers. Whenever a monitor detects a QoS flaw, an *event* will be triggered for the corresponding service handler. The quality events framework corresponds to the diagram of Figure 3.5(e).

A hybrid concept that combines feedback and feedforward is more generally applicable.

A conceptual example of a hybrid framework is presented in [Bhatti and Knight, 1999]. Feedback is given in terms of application-specific parameters. The application specifies its own so-called QoSSPACE defined as an orthogonal combination of parameters. The parameters are selected such that each mode in which the application may operate can conveniently be described as a sub-space (e.g. a cube in a 3-D parameter space). The network performance feedback is then mapped into the

QoSSpace, signalling the compatibility of each sub-space under current conditions. Since the feedback is in application-specific terms, adapting to changes amounts to selecting the mode associated with the best sub-space; see Figure 3.5(f) for a diagram.

QualMan [Nahrstedt et al., 1998] is a client/server brokerage-based platform for QoS-aware resource management. QoS contracts are negotiated based on QoS profiles that either are provided by the client or probed by the server at run time; see Figure 3.5(c) and 3.5(d) for the respective diagrams.

3.4 ARC structures: compositionality

ARC components (subsystems) are suitable building blocks for composing larger structures, which form a system again. In complex communication systems like Ubicom, many different structures coexist, some more complex than others. In this section, we consider two basic structures: a cascade structure and a parallel structure. Both structures implement distributive coordination to control shared resources. In this section, we study the consequences of the structure of the compound system. We consider their interfaces and the structure of the emerging multiobjective design problem.

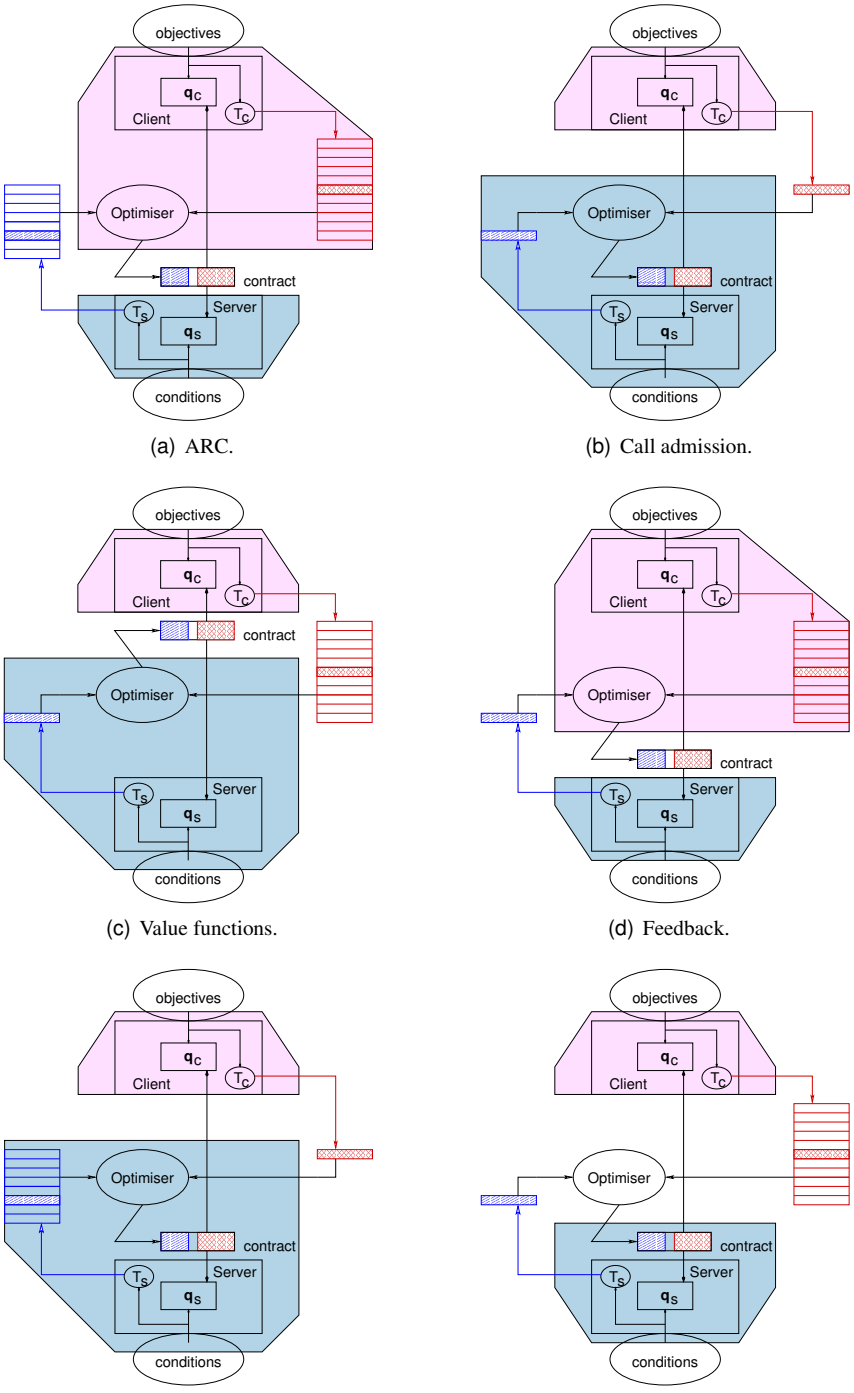
As an introduction, consider the following, abstract, example of a generic content presentation system. More practically oriented examples can be found in Chapter 6 (Section 6.1 and Section 6.2).

Example 3.4 (Content presentation) *Consider* Figure 3.6, *which shows a generic diagram of a content presentation system, which is mapped onto a generic communication system (*Figure 1.1 *on Page 2). Think of an infotainment system that offers information to a human user in an entertaining way. We presume an analysis system (not shown in the diagram) that generates an information stream U. This stream is optimally encoded through a maximum entropy coder. The consecutive synthesis system transfers an appropriate portion of U while disregarding the portion Π_U. The remaining information X is used by the actor system to present information appropriately. The actor system typically adds redundancy in order to make the information understandable for the user. Redundancy can be added in the time domain (lengthy rehearsals), in the spatial domain (vivid animations), or both. The final phase is the interpretation of the information by the user (the sensor system). The sensor is a selective device; the user only appreciates part of the offered information. Aspects that play a role in the interpretation process are the patience of the user, the ability of the user to concentrate and focus, etc. Consequently, only a part of the information as offered in Y is effectively transferred to the user V.*

The example illustrates the relation between communication systems and information theory [Shannon, 1948; Goldie and Pinch, 1991; Verdu, 1998]. The measure of information contained in a stream U is referred to as *entropy*, denoted as $H(U)$. The efficacy of a communications system depends on the ratio $H(V)/H(U)$ and the latency of the transfer (can be infinite).

Entropy is defined as an expectation [Goldie and Pinch, 1991]. Let p_i be the probability that the a symbol $\boldsymbol{u}_i$ is an element of U, then the entropy of U is defined as

$$H(U) = -E(\log_2 p_i) \tag{3.1}$$



(a) ARC.

(b) Call admission.

(c) Value functions.

(d) Feedback.

(e) Quality events.

(f) Application Adaptation functions.

Figure 3.5: QoS concepts.

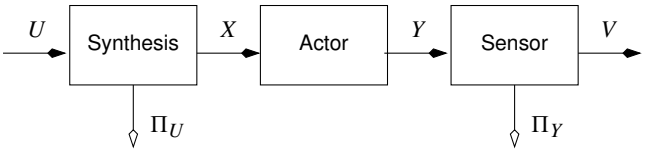


Figure 3.6: Content representation system.

The synthesis and sensor systems of Example 3.4 discard a portion of the available information. In general the mutual information of the incoming stream S and the outgoing stream T of a system is defined as

$$I(S \wedge T) = H(S) - H(S \mid T), \tag{3.2}$$

where the conditional entropy, $H(S \mid T)$, is referred to as the *equivocation* of S about T, thus the uncertainty about S when T is received.

The actor system of Example 3.4 adds redundancy so as to ease decipherability for the subsequent sensor system. Additional bits can be used to reorganise the stream X, such that deciphering of Y becomes simpler. A theoretic example is a prefix-free code, which allows deciphering without implementing lookahead. Additional bits can also be used to transfer the information multiple times, preferably in different shapes. In view of the example, think of an audio-only presentation, a video-only presentation, or a full-fledged multimedia presentation, possibly supported with 3D rendered graphics.

The content presentation system of Example 3.4 is a cascade structure when viewed from a distance. However, the description of the subsystems suggest that there are underlying parallel structures. For instance, the actor system has a choice of a wide range of displays. Optimal coding, in the sense that the user apprehends most of the information, must be entertaining in order to keep the attention of the user. This optimal coding requires close collaboration between a wide range of disciplines, ranging from multimedia coding to multimedia processing, and even to multimedia art. The artistic oriented subsystems will encapsulate the engineering oriented subsystems (symbiosis), whereas typical engineering subsystems will combine different technologies for presenting information. The latter process typically requires a minimum amount resources, but also has the danger that one display technique masks the other (epistasis).

3.4.1 Cascade structure

A cascade structure is a concatenation of components. Example 3.3 on Page 44 and Figure 3.3 contain an example of a cascade filter structure. The coordination of the shared resources, energy in the example, is organised through sharing the relevant information through the ARC (QoS) interfaces of the involved components. This resource-related information is captured in a resource vector $\boldsymbol{\rho}$. Each component in the cascade (chain) observes, and possibly updates, the resource utilisation vector. However, detailed information of a resource is only required on a need-to-know basis. From

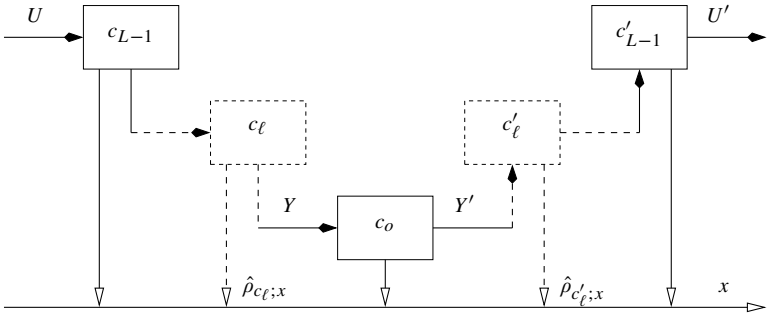


Figure 3.7: Typical cascade structure: workload on x.

the perspective of a particular component in the cascade, the resource vector $\boldsymbol{\rho}$ may bear information about engineering constructs that it is unfamiliar with. This particular entry in $\boldsymbol{\rho}$ can safely be ignored, but must be passed on. An object-oriented approach to handling the resource vector comes to mind. The object class contains the resource vector and provides specialised access methods for updating and querying the vector. This form of abstraction allows for handling of non-linear behaviour of resources and hiding of irrelevant aspects (the need-to-know basis).

A typical cascade structure is given in Figure 3.7. The layered system transfer an information stream U into U' while using the resource x. A layer ℓ applies an encoder c_ℓ and a decoder c'_ℓ that induce a load on resource x of respectively $\hat{\rho}_{c_\ell;x}$ and $\hat{\rho}_{c'_\ell;x}$. As an example, consider x being the latency. Layer ℓ regards the lower layers ($\{o \cdots \ell - 1\}$) as a non-ideal channel that also induces a workload on the resource, namely the aggregation of the individual encoders and decoders. The channel is non-ideal as layers may discard information or add redundancy.

Obviously, an ARC enabled channel can generate an offer which allows the encoder/decoder pair at level ℓ to do a tradeoff between the amount of effectively of transferred information ($I(Y \wedge Y')$ (3.2)) and the total amount of resource utilisation ($\sum_{i=0}^{\ell-1} \hat{\rho}_{c_i;x} + \hat{\rho_{c'_i}; x}$). The offered operation space of the channel provides the necessary conditions for a layer to generate an offer of its own.

A more complicated aspect of cascade structures emerges when we consider structures with dependent sources. In that case it is necessary to effectively characterise the source from one level to the other, which usually requires domain-specific expert knowledge. As an example, consider Figure 3.8. In this diagram, a source u_L (information stream $H(u_L)$) is to be coded for transport by a cascade of coders c_ℓ. A coder c_ℓ is offered a source $u_{\ell+1}$ with entropy $H(u_{\ell+1})$ of which a portion $H(\hat{y}_\ell)$ is offered for transport. The remaining $H(u_{\ell+1} \mid \hat{y}_\ell)$ is offered as a source to the remainder of the cascade, i.e., $H(u_{\ell+1} \mid \hat{y}_\ell) = H(u_\ell)$. A coder c_ℓ must know the distortion left by the remainder of cascade in order to make a proper tradeoff about which portion of the source it can encode effectively and which portion it may pass on. Let y_ℓ be the set of encoded streams from coder c_ℓ down to c_o; $y_\ell = \{\hat{y}_\ell, \hat{y}_{\ell-1}, \cdots, \hat{y}_o\}$. The distortion sought after is thus the equivocation [Shannon, 1948] of y_ℓ about u_ℓ: $H(u_\ell \mid y_\ell)$.

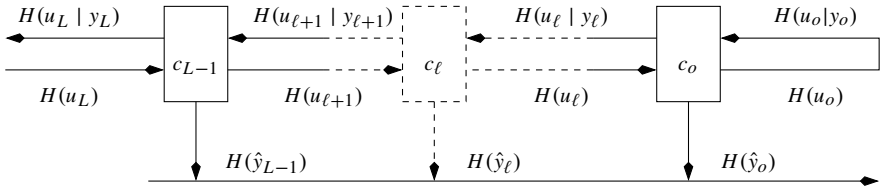


Figure 3.8: Cascade filter structure with dependent sources.

Hence the distortion as offered by the cascade as a whole equals $H(u_L \mid y_L)$.

Each system component, c_ℓ, offers a so-called rate-distortion curve. The aim is to derive the rate-distortion curve of the whole system; the aggregated bit rate of $\{H(\hat{y}_o) \cdots H(\hat{y}_\ell) \cdots H(\hat{y}_{L-1})\}$ versus, $H(u_L \mid y_L)$, the accomplished system-wide distortion . The rate-distortion curve allows a client of the compound system to trade off the bit rate (throughput) against the possible *degradation* of the source. An application of this method is proposed in [Heusdens et al., 2001] for implementing a scalable near-optimal audio codec. Each filter component implements a different strategy to encode the source information efficiently. If a cascade of these filter components is implemented, the input to the next stage obviously depends on the choices made in the previous stage. Still the system as a whole has to distribute a rate budget over the filter components.

The challenge of the system in Figure 3.8 is twofold. Individual components must be suitable to evaluate their options in a variable context. In addition, the characterisation of the dependent sources must be sound. The context of a component is identified by the imposed objectives (characterisation of the source that is offered for transport and the desired level of distortion from the preceding component in the cascade) and the conditions (characterisation of the dependent distortion by the following component in the cascade). Proper characterisation (modelling) of sources and distortion is crucial, as we observed already in Example 3.4.

3.4.2 Parallel structure

Another common filter structure is the parallel structure of Figure 3.9. There are two basic types: a *select* structure (XOR-relation) and a *merge* structure (AND-relation). The select structure chooses one service from a set of offered services, whereas the merge structure joins all services in the set. An example of a select structure is a manager of a multi-modal radio transceiver. Depending on the availability of a wireless network and the characterisation of a requested communication, an appropriate service (WiFi, GSM-900 MHz, GSM-1800 MHZ, IrDa, etc.) is invoked. All other services are shut down. The manager offers the instantaneous power-optimised solution to the application, while taking into account the application request for a specific type of communication. An example of a merge structure is the combination of multiple visual objects in one scene. Example 3.2 combined multiple video streams. One can also think of a 3D graphics scene that combines multiple graphical objects. The position of

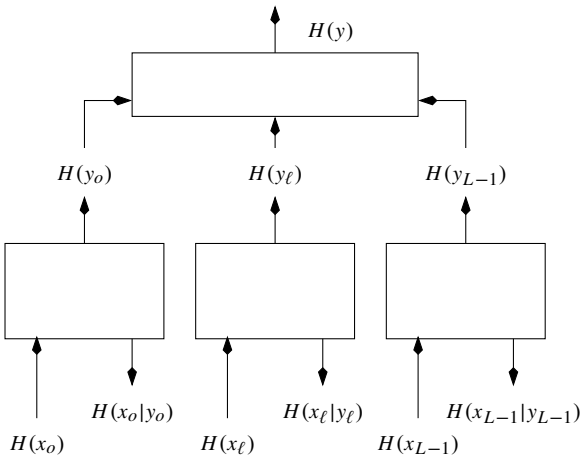


Figure 3.9: Merge filter structure.

each object (short range, medium range, or long range) determines the relative accuracy with which individual objects must be modelled [Pasman and Jansen, 2001], such that the overall scene shows a comparable distortion for all objects contained.

The select structure is a true abstraction of similar services. Each service has a specific, usually static, behaviour, performance, and resource utilisation. By combining services we sacrifice area but extend the operation space, hence incorporate flexibility, see Figure 2.12 on Page 33 for an example. Because of the similarity of the services, abstractions can be captured in a common model, which makes combining of services relatively simple.

Merge structures require sharing of information among subsystems. In a complex system with heterogeneous filter components, however, it is not a priori clear how to format this information. An authoritative framework that models the necessary information is less attractive due to inhomogeneity of the system and the requirement to support evolution.

The fundamental difference between the specification in the merge structure and the specification in the cascade structure is their interaction with their context. In the cascade structure, components utilise the shared resource directly without adaptation. The result of their individual evaluations is cooperatively propagated to the rest of the structure. In the merge structure, components adapt their evaluations to the estimated aggregated use of all other components in the structure. Their respective offers to the top-level component encompasses their cooperative intend. The top-level component guards the integrity of the aggregated resource utilisation and thus the validity of offers.

The structures of Figure 3.8 and Figure 3.9 are the result of addressing dedicated aspects of components in a particular view, namely those components that share a particular resource (engineering view) and/or implement a particular functionality (computational view).

3.5 ARC Discussion

So far, the ARC framework followed the natural course of development in a democratic setting. The coordination is established through cooperation among subsystems based on equality, rather than on dominance and subjection. The argument for such an heterarchic approach is also observed when complex systems are considered from a broader perspective.

3.5.1 Coordination and cooperation

The ARC framework we developed in this chapter and the views developed in Chapter 2 advocate a cooperative way of organising coordination. Coordination is the process of composing, regulating (the protocol), and arranging systems, with a strong emphasis on the fact that an aggregate is a system only if it has emergent properties; the whole is more than the sum of its constituents. Cooperation implies inseparability and the working together of autonomous components; "we" instead of "me" and "you". Cooperative coordination thus is distributed, shows mutual respect, and has joint responsibility for the integrity of the system. Hence subsystems necessarily are *intelligent*.

In Chapter 5 we develop a model of computation that supports autonomous processes. The model, amongst others, allows processes to be specified in an imperative language. A coordination model integrates separate activities into an ensemble [Papadopoulos and Arbab, 1998]. Here we review a selection of articles from literature that address the issue of coordination and cooperation. Quite a number of the referenced articles originate from disciplines outside the scope of complex systems (e.g. Ubicom), addressed in Section 2.2. Contributions from the field of anthropology, philosophy, and artificial-life sciences are included.

In [Malone and Crowston, 1994] coordination is defined as managing dependencies between activities. It is observed that similar coordinating constructs exist in various disciplines. In order to transport concepts from one discipline to another, we must develop a framework. Case studies from very different disciplines are evaluated. Malone and Crowston conclude, amongst others, that there is no single right way to identify components of coordination. They mention as an example the process of forcing a group decision by *authority*, majority voting, or *consensus*. It has been observed that ecosystems are often the result of heterarchic interactions rather than hierarchical interactions [Crumley, 1995]. A mathematical approach to coordination is from the field of game theory. The so-called "Prisoner's dilemma" is a non zero-sum game that offers a choice between cooperation and independent action [Salhi et al., 1996]. In Ubicom, decisions are initially by made consensus, but when time progresses and deadlines are immediate, decisions are made by authority.

An analysis of what it takes to have distant humans coordinate their actions in a cooperative way can be found in [Introna, 2001] dealing with tele-coordination and tele-cooperation. Tele-coordination is only effective when the coordinating parties share a *situated language*. That is, the parties share a mutual understanding of intentions. In ARC we observed a similar issue: component developers typically originate from different disciplines. These developers do not necessarily share a common understanding of each other's intentions. A way to overcome this problem is the use of a "situated lan-

guage", a concept originally developed by Wittgenstein in [Wittgenstein, 1963]. Here a *representational view* on language and meaning is emphasised. Depending on the situation (culture, discipline, etc.), words receive a meaning. Development of a local language (dialect) therefore is regarded a necessity rather than a complication. A suitable language supports interaction. The underlying problem is that knowledge is *tacit*, thus only available subconsciously. The articulation of tacit knowledge is tedious if not impossible; for example, try to explain in clear way how to ride a bicycle [Introna, 2001].

Introna has a case for ambiguity. Cooperation requires exchange of information so as to distinct *context* and *content*. There is however no definite way to make the distinction. Local situations require local solutions. For similar reasons we argued that a metaframework should not be defined. The following warning proves to very be true however: "discovering misunderstandings is very difficult"[Introna, 2001]. Ambiguity serves a productive role, and multidisciplinary cooperation is a definite asset in this. Different disciplines have different views, but during the process of developing a situational dialect, novel coordination methods may evolve.

The study of complex systems combines the classical computational approach and dynamic systems theory: the theory of structure and the theory of change respectively. In the ARC framework, coordination resembles a determinate computational approach and cooperation resembles the indeterminate dynamical approach. In the Chapter 5 we present a concrete example of the rapprochement of the theory of structure and the theory of change.

3.5.2 Stability

The ARC framework is based on consensus, which is achieved through negotiations. A recurring issue of this type of coordination frameworks is their stability. Stability is not guaranteed. In fact instability is lurking in every closed-loop control system [see e.g. Dorf, 1989]. Many theoretical and practical applicable results have been derived for linear systems with feedback control. A number of important parameters influence the stability of the resulting controlled system. Variations in the *gain* parameter, the closed-loop gain, and the transient gain affect the relative stability of a system. The *frequency* response of a system determines the stability under excitation of time-varying input. Known *disturbance* can be counteracted by taking appropriate measures. A particular form of disturbance is *delay* or *lag*. The measurements from a sensor, or the actions of a control system, may lag behind. When this is known, appropriate measures can be taken. Otherwise this disturbance increases instability. In Section 6.2 we present a case study that incorporates lag of sensor data.

With respect to the ARC framework we observe that components typically do not satisfy the linear system conditions: superposition and homogeneity. On the other hand we know from operational research that a less stringent property, *monotonicity*, suffices to locate an optimum. Combining these observations we conjecture that instability can be prevented in the ARC framework when

- operation spaces are monotone,
- the time scale of components increases with the level (index) in the composition.

In practical situations, components have often monotone operation spaces. Operation points can be located such that the abstraction of behaviour, capacity, and resource utilisation is monotone. One may argue that monotonicity is the result of being cooperative. It is very hard to reason about non-monotone components.

Oscillations and possible instability, frequently occur when a system is iteratively controlled. However, when operation spaces are monotone, the coordination process can locate the equilibrium. In the compositional structures, as considered in Section 3.4, components apply services from lower-level components, maintaining monotonicity implies an increase of the time scale and an increase of the impact of components higher up the structure. In case of a mobile video communication system (Example 3.3 on Page 44) the impact of the video encoder is larger than the impact of the channel modulator. If the video encoder switches operation from 30 frames per second (FPS) to 8 FPS, this is highly noticeable. If the channel modulator switches from 256 quadrature amplitude modulation (QAM) to 16 QAM on the other hand, the resulting throughput changes dramatically, but the impact of this switch may be hidden by the rest of the system. However switches from 256 QAM to 16 QAM can occur more frequently, at a significantly smaller time scale, than switches from 30 to 8 FPS, given the same relative overhead.

The ARC framework allows for ambiguity in the sense that there is no metaframework. Ambiguities, if any, must be resolved locally. The underlying paradigm of keeping communications local has been articulated in various places in literature. In case of ARC, the main reason for local communications is the support of evolution. Satyanarayanan advocates local communications because of *scalability* of a design. In [Lieberherr et al., 2001] a run-time implementation of aspect-oriented programming is described that obeys "The law of Demeter": Objects should only have knowledge of closely related objects.

The balance between the volume of an ARC interface and global optimisations has similar considerations as partitioning a complex system in its constituent components. It is generally accepted that components must be substantial yet not too complicated. The same applies to ARC interfaces: a practical limit of ARC interface parameters is 4, which we found to be suitable for many practical situations [Taal et al., 2002; van Dijk et al., 2000a; Pasman and Jansen, 2001].

There are a number of situations in which parties feel the need for more than these 4 interface parameters. For example, if the underlying optimisation problem of a component can be very complex then partitioning is a logical step. A canonical (minimal) coding of the information that is to be communicated yields a minimum number of (near) orthogonal parameters. However, in the occuring situation the optimisation problem became less complex by introducing redundancy in the information model.

An operation space may be implicitly structured. In occurring cases, a server and a client interpret the operation space similarly (c.f. the manifold of Section 2.1.1). In that case, they assign equal, implicit, dependencies to parameters in the operation space, which circumvents the use of additional parameter that make the implicit interpretation explicit. An example of such an efficiency operation can be found in [Taal et al., 2002].

Designing a proper ARC interface is not a trivial task. Obviously the client lacks information which the server may be able to provide and vice versa. The goal is to find a proper tradeoff. Not only the information must be cast into a clear format, but

it must also fulfil the need for information. Overprovisioning is a waste. Recent advances on collaborative sensor networks [Kumar et al., 2002; Zhao et al., 2002] take an information theoretic approach, which is arguably less suited towards practical implementations, but certainly an interesting angle.

3.6 ARC components

ARC components are context aware and cooperative. They supply and employ services of which the behaviour is established through negotiated (QoS) contracts. Clients are serviced through their server-side interface, and servers are employed through their client-side interface. At the same time, component often have uncoordinated input and output relations as well. Example 3.3 on Page 44 showed an example of an uncoordinated input relation. There the characteristics of a video source are monitored (sensed), as they contribute to the context of a component. An example of an uncoordinated output relation involves a component that is allowed to employ services from a CPU without establishing a contract.

Whether or not a component will monitor the performance of employed services is a matter of *trust*. Likewise, the policing [Rathgeb, 1991] of the induced workload from serviced clients is also a matter of trust. In a true democratically coordinated communicating system, trust is a matter of course. A true democratic system thus circumvents the need to implement tedious monitoring and policing instruments and gains efficiency. In the sequel, we ignore any out-of-contract performance or workload numbers. Obviously, if a component uses the input from an uncoordinated interface, monitoring is required.

Being cooperative, the developer is well aware of the fact that a certain behaviour of the component may influence the rest of the system in an unspecified way. A conceptual diagram of an ARC component is given in Figure 3.10. The Figure presents a jigsaw piece, indicating that the component is part of a larger system of cooperating and communicating components, e.g. compare it with Figure 2.3 on Page 20. The component in Figure 3.10 also encompasses a generic communication system from sensor to actor, e.g., compare it with Figure 1.1 on Page 2. Compositionality of an ARC component is due to its context awareness, which is implemented by a local coordinating entity that autonomously takes decisions. In Chapter 5 this entity is referred to as an *oracle*.

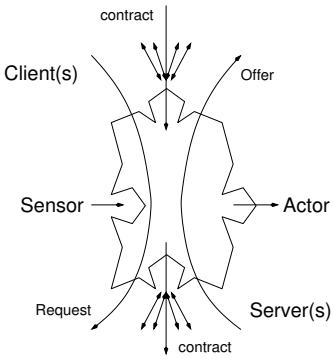


Figure 3.10: ARC context-aware component.

Four

Mathematical consideration of QoS

THE ARC framework of the previous of the chapter defines an optimisation problem. The milieu of a system component defines the (local) context in which the component evaluates its optimisation problem. Internal decision variables must be selected such that imposed objectives are met and that imposed conditions are not violated. The optimisation problem is in essence a multiobjective or vector optimisation problem, which in general requires expert knowledge to solve it successfully.

In this chapter we introduce a mathematical interpretation for the emerging optimisation problem. We refer to related work (Section 4.2) for an overview of available technology to evaluate the optimisation problem. In Section 4.3 we analyse the consequences for the optimisation problem when components are used in composition. This chapter is concluded with a case study, which demonstrates that even in a simple setting multiobjective optimisation problems are tedious. A fully mathematical implementation of ARC therefore is not feasible for practical reasons.

4.1 Vector problem definition

ARC components solve a multiobjective optimisation problem. In general, a multiobjective optimisation problem [Haimes et al., 1989] (or vector optimisation problem) is defined as in (4.1), where $\boldsymbol{f}(\boldsymbol{x})$ is the objective function and $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ partition the solution space in a feasible and an infeasible part. $\boldsymbol{g}(\boldsymbol{x})$ identifies a half-space, whereas $\boldsymbol{h}(\boldsymbol{x})$ identifies a subspace. Note that $\boldsymbol{h}(\boldsymbol{x})$ is a convenience function that could have been included in $\boldsymbol{g}(\boldsymbol{x})$.

$$\begin{aligned} \min_{\boldsymbol{x}\in X} \boldsymbol{y} = \min_{\boldsymbol{x}\in X} \boldsymbol{f}(\boldsymbol{x}) = \min_{\boldsymbol{x}\in X} \{f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x})\} \\ \text{subject to,} \quad X = \{\boldsymbol{x} \,|\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\} \end{aligned} \tag{4.1}$$

where,

$$\begin{aligned} \boldsymbol{y} &= m\text{-dimensional objective vector,} \\ \boldsymbol{x} &= n\text{-dimensional vector of decision variables,} \\ X &= \text{the parameter space; the set of feasible solutions.} \end{aligned}$$

The objective vector $\boldsymbol{y}$ contains the set of non-inferior or Pareto points. A decision $\boldsymbol{x}^\star$ is said to be an non-inferior solution of (4.1) if and only if there does **not** exist another $\bar{\boldsymbol{x}}$ such that $f_j(\bar{\boldsymbol{x}}) \leq f_j(\boldsymbol{x}^\star), \forall j$, with strict inequality for at least one j. Note that practical implementations offer a subsampled set of Pareto points [Morse, 1980; Rosenman and Gero, 1985]. The ARC framework is a mediator. It provides concepts, but does not prescribe a generic methodology for solving the emerging nonlinear multiobjective optimisation problems. There exist many solution methods for these types of problems, but a universal solution has not yet been found. The chosen solution method is in general very domain dependent. A recent overview of approaches to multiobjective optimisation problems can be found in [Miettinen, 1999].

With respect to Figure 3.10 on Page 60, an ARC component approaches the multiobjective optimisation problem in two phases. The first phase is executed after a *request* has been received from the client and results in a request to any of the employed services. Since requests define a partial context, the first phase boils down to a partial evaluation of the underlying optimisation problem. The second phase starts after the reception of *offers* from all employed services. At this stage the context is fully determined and thus allows for an evaluation of the full optimisation problem. The resulting solution yields the offer that will be issued to the client of the ARC component.

Figure 4.1 specifies the parameters involved in the multiobjective optimisation problem for ARC components. The vectors $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{y}}$ are the incoming and the generated requests, respectively. Similarly, the vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ are the received and the generated offers. The vector $\boldsymbol{w}$ is the observed, uncoordinated, sensor data. The actor data, although explicitly assigned to a vector $\boldsymbol{v}$ in the diagram, will be mostly ignored in the rest of this section. We regard $\boldsymbol{v}$ as *noise*, which interferes with the rest of the system. The internal decision parameters of the component are gathered in the vector $\boldsymbol{q}$. Parameters from $\boldsymbol{q}$ refer in part to algorithmic options and in part to choices of how computational constructs are mapped to engineering constructs. Not every utilisation of engineering constructs (resources) will be contracted and thus accounted for in $\tilde{\boldsymbol{y}}$ and $\boldsymbol{u}$. During system design, one may decide to utilise a resource without informing the rest of the system, so implicitly through the actor interface $\boldsymbol{v}$. As long as the noise is within bounds, efficiency is gained while the system integrity is not jeopardised. A typical example is the uncoordinated use of a CPU.

The set of named vectors of Figure 4.1 *abstracts* the behaviour, performance, and resource utilisation of a component. Separation of concerns [Ossher and Tarr, 2001] of the set into subsets emphasises the distributive aspect of ARC. We address the *adaption* and *cooperation* aspects of the component. The sensor data, $\boldsymbol{w}$, provides information necessary for successful adaptation to a changed environment. The offered operation space $\boldsymbol{u}$ also contributes to the adaptation process. Cooperation of the component with its immediate environment is achieved through making options explicit through the generated offer $\boldsymbol{y}$. The actor output, $\boldsymbol{v}$, potentially provides information to the environ-

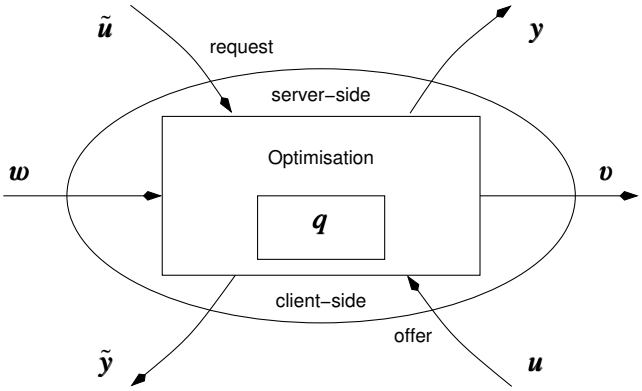


Figure 4.1: ARC component parameter interface.

ment in a cooperative way, if it is made explicit. The incoming ($\tilde{\boldsymbol{u}}$) and outgoing ($\tilde{\boldsymbol{y}}$) requests facilitate efficient communication between components.

Operation spaces at either side of an ARC component have natural partitions. Both $\boldsymbol{y}$ and $\boldsymbol{u}$ are tuples of distortion, capacity, and resource utilisation: $\langle \boldsymbol{y}_\delta, \boldsymbol{y}_\tau, \boldsymbol{y}_\rho \rangle$ and $\langle \boldsymbol{u}_\delta, \boldsymbol{u}_\tau, \boldsymbol{u}_\rho \rangle$, respectively. As indicated previously, the emerging multiobjective optimisation problem is evaluated in two stages. The first stage transforms server-side requests ($\tilde{\boldsymbol{u}}$) into client-side requests ($\tilde{\boldsymbol{y}}$). The second stage evaluates the fully defined multiobjective optimisation problem, given in (4.2).

We will first analyse the fully defined problem of the second stage and then return to the problem of the first stage, which can be seen as a partial definition of the problem definition of the second stage.

$$\begin{aligned} \min_{(\boldsymbol{q},\boldsymbol{u})} \boldsymbol{y} = \min_{(\boldsymbol{q},\boldsymbol{u})} \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u},\boldsymbol{w}) = \min_{(\boldsymbol{q},\boldsymbol{u})} \left\{ \boldsymbol{f}_\delta(\boldsymbol{q},\boldsymbol{u},\boldsymbol{w}), \boldsymbol{f}_\tau(\boldsymbol{q},\boldsymbol{u},\boldsymbol{w}), \boldsymbol{f}_\rho(\boldsymbol{q},\boldsymbol{u},\boldsymbol{w}) \right\} \\ \text{subject to,} \quad \left\{ (\boldsymbol{q},\boldsymbol{u}) \,\middle|\, \boldsymbol{g}(\boldsymbol{q},\boldsymbol{u},\tilde{\boldsymbol{u}},\boldsymbol{w}) \leq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},\tilde{\boldsymbol{u}},\boldsymbol{w}) = \boldsymbol{0} \right\} \end{aligned} \tag{4.2}$$

The decision variables, $\boldsymbol{x}$ in (4.1), have been partitioned in (4.2). The optimisation algorithm must decide on the internal decision parameters $\boldsymbol{q}$ and select an entry from the offered operation space $\boldsymbol{u}$. The objective functions $\boldsymbol{f}(\cdot)$ also take into account uncontrolled observations $\boldsymbol{w}$. The feasible solution space X of (4.1) is specified in (4.2) by the functions $\boldsymbol{g}(\cdot)$ and $\boldsymbol{h}(\cdot)$. These functions can be decomposed into functions $\boldsymbol{g}^\gamma(\cdot) \subset \boldsymbol{g}(\cdot)$ and $\boldsymbol{h}^\gamma(\cdot) \subset \boldsymbol{h}(\cdot)$, with $\gamma \in \{a, b, \cdots\}$. The subfunctions $\boldsymbol{g}^\gamma(\cdot)$ and $\boldsymbol{h}^\gamma(\cdot)$ address specific concerns. Each concern corresponds to specific constraints on incoming requests $\tilde{\boldsymbol{u}}$ and offered options $\boldsymbol{u}$. What follows is an enumeration of specific concerns. Implementation of these concerns are exemplified in Chapter 6.

Decision space Individual decision parameters are bound by the internal process. These bounds usually have a physical origin. The range of a variable capacitor, for example, is simply limited by the applied technology. The feasible range of settings for each

decision variable is identified by a subset of $\boldsymbol{g}(\cdot)$ and $\boldsymbol{h}(\cdot)$, denoted as $\boldsymbol{g}^a(\cdot)$ and $\boldsymbol{h}^a(\cdot)$, respectively. These subsets of functions are independent of the context of the component. The feasible range of settings for decision variables, limited by the decision space, is specified in (4.3).

$$\left\{\boldsymbol{q} \,\middle|\, \boldsymbol{g}^a(\boldsymbol{q}) \leq \boldsymbol{0}, \boldsymbol{h}^a(\boldsymbol{q}) = \boldsymbol{0}\right\} \tag{4.3}$$

Consistency The causal effect of setting a decision variable must be consistent with the context of the component. The context of a component is partly determined by the incoming request, $\tilde{\boldsymbol{u}}$, and partly by the incoming offer, $\boldsymbol{u}$.

Incoming requests are specified as half spaces. In practical situations, requests often specify a region bound by two parallel planes: upper and lower bounds. The subset $\boldsymbol{g}^b(\cdot)$ of $\boldsymbol{g}(\cdot)$ defines the range of settings of decision variables such that consistency is guaranteed. The range is specified in (4.4).

$$\left\{\boldsymbol{q} \,\middle|\, \boldsymbol{g}^b(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{w}, \tilde{\boldsymbol{u}}) \leq \boldsymbol{0}\right\} \tag{4.4}$$

Offers received from employed services specify the optional behaviour and performance of these services. A fully determined operation point, however, also specifies the workload imposed by the client on the server. The consistency of a component with an offer $\boldsymbol{u}$ can be expressed through a subset $\boldsymbol{h}^b(\cdot)$ of $\boldsymbol{h}(\cdot)$. The corresponding discrete range of settings of decision variables is specified in (4.5).

$$\left\{\boldsymbol{q} \,\middle|\, \boldsymbol{h}^b(\boldsymbol{q}, \boldsymbol{w}, \boldsymbol{u}) = \boldsymbol{0}\right\} \tag{4.5}$$

Compatibility In practical situations, some resources are coordinated in an authoritative manner from a central entity, particularly those resources that are less sensitive to interference, often because of a surplus of available resources. The usage of these resources is accounted for in the noise vector $\boldsymbol{v}$. In occurring situations, the central coordinating entity will assign a budget $\hat{\boldsymbol{\rho}}$ to individual components for uncoordinated use. The effect of selecting decision variables must be compatible with any assigned budgets. As before, a subset $\boldsymbol{g}^c(\cdot)$ of $\boldsymbol{g}(\cdot)$ can be constructed for this purpose. The resulting range of settings of decision variables is specified in (4.6).

$$\left\{\boldsymbol{q} \,\middle|\, \boldsymbol{v} = \boldsymbol{g}^c(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{w}, \hat{\boldsymbol{\rho}}) \leq \boldsymbol{0}\right\} \tag{4.6}$$

Above, we outlined the two-stage evaluation process of the optimisation problem from (4.2). The first stage (request) generates $\tilde{\boldsymbol{y}}$ and the second stage (offer) generates $\boldsymbol{y}$. Let $\hat{\boldsymbol{u}}$ denote a relaxation (rough approximation) of $\boldsymbol{u}$. The eventual offer $\boldsymbol{u}$ has precise operation points, whereas $\tilde{\boldsymbol{u}}$ relaxes each operation point to an operation vicinity. Typically vicinities coalescence to the operation area covered by the eventual offer. The goal is to recover the boundaries of the operation area of all offers such that the constraints (4.3) through (4.6) are met, with $\boldsymbol{u}$ substituted by $\hat{\boldsymbol{u}}$. Subsequent evaluation of (4.2) yields the boundaries of the operation area, hence it yields $\hat{\boldsymbol{u}}$. The request $\tilde{\boldsymbol{y}}$ then simply boils down to putting $\tilde{\boldsymbol{y}} = \hat{\boldsymbol{u}}$. Note that the derivation of $\hat{\boldsymbol{u}}$ from the constraints is not a trivial task. The consistency constraint (4.4), for example, requires the

existence of the inverse of $\boldsymbol{g}^b(\cdot)$ with respect to $\boldsymbol{u}$, or at least a close approximation. Modelling of performance, behaviour, and resource utilisation therefore is crucial for true context-aware components.

4.2 Related Work

In this section we give an overview of the vast amount of literature on the subject multiobjective and multidisciplinary optimisation. The references in this section might prove useful for analysis and run-time implementations of concepts laid out in this chapter.

Individual aspects of multiobjective hierarchical multidisciplinary design optimisation problems can be found in literature. It is interesting to notice that references originate from a wide range of disciplines, which sometimes differ widely from the disciplines involved in a communicating system design. Communicating systems have multilevel structures (with nests) as a natural organisation. Multiobjective optimisation is an upcoming research issue for mono-disciplinary or at best oligo-disciplinary components in communicating system design. Multidisciplinary design optimisation (MDO) and multidisciplinary design analysis (MDA) are emerging methodologies in hard engineering disciplines like mechanical engineering and aeronautics [Livne, 1999]. Unfortunately they usually address a single objective and lack the notion of a multilevel system.

Our method of decomposing a large-scale multiobjective optimisation problem is indirectly related to emergent behaviour analysis. The main reason for this is that we use aspect decomposition of the communicating system [Papalambros and Michelena, 2000]; see also Section 2.2. Control theory [Stoilov and Stoilova, 1999] is an example of a discipline that applies similar approaches.

4.2.1 Multiobjective optimisation

The formulation of multiobjective optimisation problems dates back to the 1960s. Only recently applications have emerged in the area of design for embedded systems. Most projects, however, address mono-disciplinary research. A distinct example is HP's program in chip out (PICO) project. In [Schreiber et al., 2000] a methodology is presented that explores the design space of a network of non-programmable processor elements. The methodology is based on combining the Pareto operations spaces of the constituent components. The underlying method is explained in [Abraham and Rau, 2000]. The necessary control implementation is described in [Aditya and Rau, 2000].

Recent publications demonstrate multiobjective design criteria for individual components. Zitzler and Thiele use genetic algorithms, see [Zitzler and Thiele, 1999] and references therein, to explore the design space of implementing a H.263 video decoder. A tradeoff is made between real-time performance and resource utilisation. Numerous other examples exist, but always with a single optimisation problem formulation and methods to solve the problem. Noteworthy examples are MOSES [Coello Coello and Christiansen, 1999], a tool for engineering optimal designs and [Mottahed and Manoochehri, 2000], who address a multidisciplinary approach for optimal packaging

of electronic systems. Yet another discipline is power load management in electric distribution networks [Jorge et al., 2000]. Notwithstanding its use of genetic algorithms, multiobjective optimisation has gained recent research interest, with promising results [Coello Coello, 1999].

Value functions typically define a single objective optimisation problem, while a multiobjective optimisation problem is expected. QoS addresses multiple concerns and is based on multiple parameters (dimensions). Implicit in most value functions is an ordering through weighing of objectives; in multiobjective optimisation theory this is referred to as the utility function. A utility function effectively maps the multidimensional objective space to a one-dimensional space. Structured methods to accomplish this not so trivial task exist. The analytical hierarchy process (AHP) [Saaty, 1990] is explicitly mentioned in [Lee et al., 1999a]; another example is Pareto race [Karaivanova et al., 1995].

Unfortunately, in complex systems the weights that effectively order the objectives are dynamic too, which is especially true for those systems that are in a constant state of flux. Value functions are not particularly suited to handle this kind of dynamics. Consider a service that uses a value function with a fixed utility function. As the service does its optimisations based on the value function, the service is considered a *best-effort* service. Ergo, best-effort services are not good enough for being used for systems in flux.

4.2.2 Multidisciplinary optimisation

There exist several formulations of problems in multidisciplinary design and accompanying methods for solving these problems [Alexandrov and Lewis, 2000a,b]. With respect to our problem formulation, collaborative optimisation [Braun and Kroo, 1997; Kroo and Manning, 2000] is the most attractive one. Collaborative optimisation is a bilevel decomposition of a large optimisation problem. The system level (level 0) optimises the system-wide objective. It issues design targets to the subsystems (constituent disciplines) at level 1. The subsystems form an optimisation program that optimises the discrepancy of the inter-disciplinary consistency constraints. The complement of collaborative optimisation is the decomposition method called " optimisation by linear decomposition": it maintains interdisciplinary consistency at system level and minimises violations of the disciplinary design constraints at subsystem level. An illustrative example of collaborative optimisation with a disciplinary design alternative (conceptual design) is presented in [Balling and Rawlings, 2000].

In [Kim et al., 2000] a system with more than two hierarchical levels is presented. Unfortunately the system does not adapt to changing environment conditions; it is merely a top-down approach. In [Tappeta et al., 2000] a design strategy is developed supporting multiple objectives and incorporating multiple disciplines.

In [Sobieszczanski et al., 1998; Kodiyalam and Sobieszczanski, 2000] a method called bilevel integrated system synthesis (BLISS) is introduced. The authors claim that problems tend to grow prohibitively large during system synthesis since collaborative optimisation combines design optimisation and design analysis. "Ultimately one needs both domain-exploring methods, and path-building methods". BLISS provides such a framework by switching between two different perspectives when appropriate;

BLISS/A evaluates the partial derivatives of the coordination variables versus the system variables, whereas BLISS/B applies Lagrange multipliers for doing the coordination.

Another noteworthy effort is the design of an co-evolutionary architecture [Nair and Keane, 1999, 2000] for distributed optimisation of complex coupled systems. Although this is in essence a bilevel optimisation procedure, it is appealing. Due to its usage of evolutionary algorithms, it circumvents precise mathematical functions by not relying on any gradient method.

The aforereferenced multidisciplinary approaches all have in common that they start from the general optimisation problem and solve it in a distributed, possibly concurrent manner, whereas in our approach the general optimisation problem is the result of analysis afterwards, after the system has been structured and connected. The coordination methods that emerge from the ARC framework share with the aforementioned methods that domain experts keep their autonomy in the design process. This is the crucial factor in acceptance of any multidisciplinary design optimisation method.

4.3 ARC structures: compositionality

Similar to our contemplation in Section 3.4 we consider in this section compositions of system components. We are especially concerned with the consequences on the underlying optimisation problem.

4.3.1 Cascade structures

The cascade structure is a concatenation of components. In this section we study, by means of an example, the consequences of this structure for the individual components.

Consider a system that applies the ARC framework to coordinate the utilisation of a particular resource x (Figure 4.2). The load of a component c_ℓ ($\ell \in \{0, 1, \cdots, L-1\}$) on resource x in a cascade $c_o \cdots c_{L-1}$ is given by $\hat{\rho}_{\ell;x}$. Let $\rho_{\ell;x}$ be the partial sum of the workloads up to ℓ, with $\rho_{-1;x} = 0$, thus

$$\rho_{\ell;x} = \sum_{i=0}^{\ell} \hat{\rho}_{i;x} \quad .$$
$$\rho_{\ell;x} = \hat{\rho}_{\ell;x} + \rho_{\ell-1;x}$$

We define the sum of all partial workloads as $\rho_x = \rho_{L-1;x}$. The cascade structure propagates the partial sum $\rho_{\ell;x}$. At the top level, component c_{L-1} selects the initial contract and therefore determines (implicitly) the distribution of the power budget (and the distribution of x) over the components in the cascade.
Suppose the resource x has a power dissipation relation

$$P(\rho_x) = (\rho_x)^\alpha \tag{4.7}$$

for some $\alpha > 0$. A trivial power relation is a linear relation ($\alpha = 1$). Then the value and the update of ρ_x^ℓ is simple. Each component c_ℓ specifies the amount of power it dissipates through $P(\hat{\rho}_{\ell;x})$ and propagates the partial sum; $\rho_{\ell;x} = P(\hat{\rho}_{\ell;x}) + \rho_{\ell-1;x}$. In

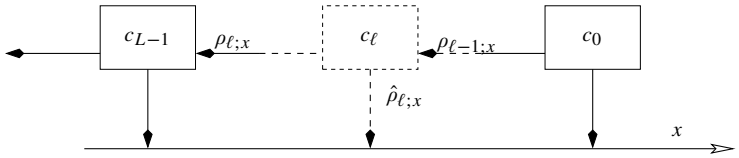


Figure 4.2: Cascade structure: workload on x.

most cases, however, the power-workload relation is nonlinear. The power dissipation of an integrated circuit with optimal supply voltage, for instance, is proportional to the cubed clock frequency ($\alpha = 3$). In that case the propagation of the partial power dissipations is suboptimal. The problem is that components down the cascade have "cheaper" energy than components up the cascade. Consequently, components down the cascade will make a different tradeoff than components up the cascade structure, simply because components down the cascade have a smaller offset. The actual power dissipation for a component c_ℓ amounts to:

$$\begin{aligned} P(\hat{\rho}_{\ell;x}) &= P(\rho_{\ell;x}) - P(\rho_{\ell-1;x}) \\ &= P(\rho_{\ell-1;x} + \hat{\rho}_{\ell;x}) - P(\rho_{\ell-1;x}). \\ &= \left(\rho_{\ell-1;x} + \hat{\rho}_{\ell;x}\right)^\alpha - \left(\rho_{\ell-1;x}\right)^\alpha \end{aligned} \tag{4.8}$$

With $\alpha = 1$, (4.8) reduces to $P(\hat{\rho}_{\ell;x}) = \hat{\rho}_{\ell;x}$ as we saw before. But with $\alpha = 2$, (4.8) expands to $P(\hat{\rho}_{\ell;x}) = (\hat{\rho}_{\ell;x})^2 + 2\,\hat{\rho}_{\ell;x}\,\rho_{\ell-1;x}$. The second (offset) term increases with ℓ. For $\alpha > 2$ the offset term increases even more rapidly with the level of components.

In the ideal situation it is possible to evaluate the relative performance of all components in the structure. Thus all components have a similar context, in this case a similar offset. There is a straightforward way to accomplish this. From the perspective of component c_{L-1} the exact distribution of the available resource budget is irrelevant. It is the same whether c_{ℓ_1} or c_{ℓ_2} consumes the better half of the budget; what matters is the total resource utilisation ρ_x. Instead of propagating the power function evaluation, one propagates a specification of $\boldsymbol{\rho}$. As a result, individual components can, as before, add their partial resource usage to the running sum. Only the top-level component will evaluate the power dissipation function, i.e., (4.7). Individual components do their optimisation in a void context, that is, apply $P(\hat{\rho}_{\ell;x})$ directly without any offset. As an example, in Example 3.3 on Page 44 the propagated resource utilisation is in cycles per second; only the top level component evaluates the power dissipation. Note that in a system with a CPU that supports frequency and voltage scaling, the top-level component sets the system wide target, and thus becomes the only explicit (contracted) user of the resource.

4.3.2 Parallel structures

A commonly applied scheme for partitioning of an optimisation problem in a parallel structure is illustrated in Figure 4.3. In this diagram two levels are recognised: a

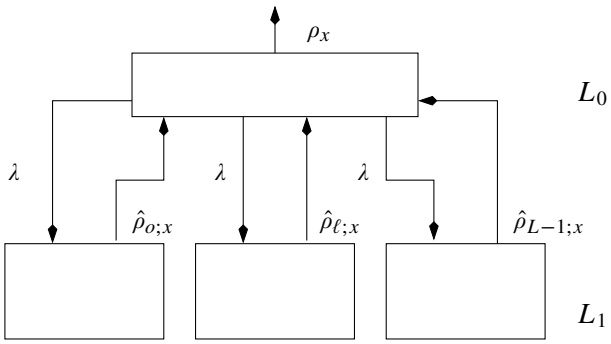


Figure 4.3: Parallel optimisation structure.

top-level, L_o, and a set of independent filter components c_ℓ at level L_1. The context in which each filter component does its optimisations is controlled by the top level component. Typical schemes require a number of iterations to find the global optimum (a single operation point). Using the ARC scheme, we circumvent iterations through the use of multiobjective operation spaces rather than of a single operation point. The abstraction of the system as a whole is done by the top-level component, based on information provided by the filter components at level L_1.

In [van der Schaaf et al., 2000], a typical optimisation scheme is discussed for an application with homogeneous filter components: a merge of identical video coders. The concurrent video coders are coordinated through the application layer L_o. The global constrained optimisation problem is partitioned and distributed over the filter components. The coordinating component determines the optimal so-called Lagrange multiplier λ. The Lagrange multiplier[1] carries the shared information for all filter components. Given λ, individual filter components solve their rate-distortion problem in a unique way. The result $(\hat{\rho}_{\ell;x})$ is fed back to the top level. If necessary the top level component adjusts the λ factor – effectively distributing the resource budget – and invokes a new iteration. The scheme assures that the individual filter components do their optimisations in a common context. In particular the collective of components arrives at operation points that expose similar distortion for each video stream, while controlling the cumulative bit rate (resource utilisation).

For a parallel structure with heterogeneous filter components, the operation spaces of the individual components typically do not nicely match. Distribution of a Lagrange multiplier therefore is nor straightforward nor generic. Yet a very similar approach is feasible. Recall the problem of distributing a resource x over the filter components. In Section 3.4.1 we distributed the power model (4.7) through its specification $\rho_{\ell;x}$ and let the top-level component c_{L-1} make the eventual translation to the absolute power figure. Here we pursue a similar approach but take the specification one step further. Usually it is not an issue where resources are spent as long as their sum does not exceed

[1] See Section 4.4 for an application of Lagrange multiplier theory.

a certain limit. Suppose the parallel structure has a budget of ρ_x, then the total power dissipation is given by $P(\rho_x)$. A linearised (fair) power dissipation model thus is

$$\begin{aligned} P_{\text{lin}}(\hat{\rho}_{\ell;x}) &= \omega\, \hat{\rho}_{\ell;x} \\ &= \frac{P(\rho_x)}{\rho_x} \hat{\rho}_{\ell;x} \quad . \end{aligned} \tag{4.9}$$

As in the homogeneous parallel case, the factor ω is broadcasted over the filter components. The factor ω provides the components with the necessary context. The resulting partial power figures are offered to the top level coordinator. The coordinator must perform a consistency check to assure that the cumulative sum of the partial power figures does not exceed the proposed budget ($P(\rho_x)$). The received offers in the ARC framework contain a range of possible operation points, based on the request ω. The subsequent optimisation problem at level L_o has the nature of a binary perfect matching problem, which includes a consistency check. The perfect matching problem is a so-called $0-1$ combinatorial optimisation problem [Nemhauser and Wolsey, 1988]. It optimises the selection from each of the offered operation spaces under constraint that the total budget matches ω. Note that since ARC allows the components of a system to request offers for a range of budgets, thus for a range for ω in (4.9), the number of iterations is confined. The resulting matching problem, however, increases in dimension and consequently becomes more complicated.

4.4 Case study

In this section, we demonstrate how operation spaces are generated within the ARC framework. We show that even in this carefully casted case, solving the optimisation problem involves tedious formula manipulations. The offered operation spaces contain a set of non-inferior or Pareto-optimum points. These operation spaces are typically multidimensional, offering a tradeoff between, for instance, performance and resource utilisation. The underlying optimisation problem, consequently, is a multiobjective one. In rare cases an operation space can be derived analytically. A closed expression of the operation space offered by a server is a necessary, however an insufficient condition for a client to construct an analytical operation space, effectively implementing functional compositionality.

Consider the two-level network of Figure 4.4, with levels 0 and 1. Level 0 has two processes X^a and X^b, which offer services to a process Z at level 1. Process Z effectively *merges* the operation spaces of X^a and X^b and offers a system wide operation space to the user of the system (not shown). Think for example of an operation space that offers the user a tradeoff between a resource utilisation (CPU) and a corresponding distortion of the processed signal. Each point in the operation space as offered by Z, and thus by the system, corresponds to an instantiation of X^a and X^b. We take the system wide resource utilisation simply to be the sum of those offered by the selected operation points of X^a and X^b, $\rho_{\text{system}} = \rho_{X^a} + \rho_{X^b}$. The system wide distortion is taken a (balanced) mean of those offered by X^a and X^b. Although process Z is aware of the CPU it does not use the CPU directly. The actual processing of the signal is done by the processes X^a and X^b, which requires the use of the CPU.

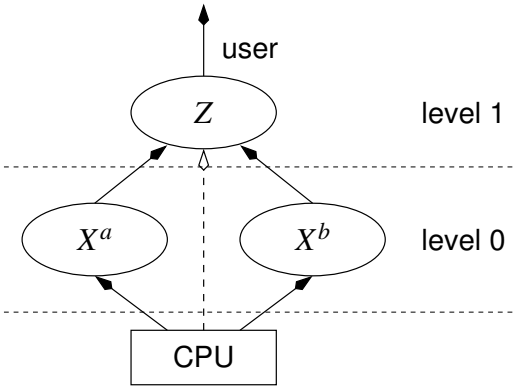


Figure 4.4: Distributed shared resources.

If process Z is to derive an operation space analytically then the processes X^a and X^b must generate a non-inferior solution of a bi-objective optimisation problem. Inspired by [Zitzler and Thiele, 1999] we use, as an example, a generalised version of Schaffer's F_2 function to model the distortion and power dissipation measures of these server processes. Schaffer's F_2 function models a quadratic relation between the distortion and the power dissipation given an internal control value. This is fairly intuitive modelling, practical examples may typically yield quadratic or even cubic relations.

Process Z has to transforms an in essence a bi-objective optimisation problem into a single objective (nonlinear) optimisation problem. To do so, Z applies a so-called *utility function*. Given a maximum total resource utilisation ρ, Z must optimally combine offers from X^a and X^b under the constraint that the maximum allowed total resource utilisation is not exceeded. Optimality here refers to a minimised distortion. A typical scenario involves a level 1 process Z that merges multiple objects from level 0 processes into a single compound object. An example of such a system can be found in Section 6.2.1. The emerging – system wide – optimisation problem is an hierarchically structured non-linear multiobjective one [Haimes et al., 1989; Miettinen, 1999], which in this particular case can be solved analytically.

4.4.1 Multiobjective optimisation

With respect to Figure 4.4, processes X^a and X^b have to generate the Pareto-optimum solution of a bi-objective or vector optimisation problem. In this section we derive the analytical solution to a generalised Schaffer $\boldsymbol{f}_2$ problem.

Let $F_2(t; c)$ define a parameterised quadratic function[2] of variable t and the (con-

[2]The notation for $F_2(t; c)$ is taken from [Haimes et al., 1989].

stant) parameter c.

$$F_2(t; c) = (t - c)^2 \tag{4.10}$$

Think of $F_2(t; c)$ specifying the resource utilisation as a function of an internal decision parameter t and a constant design parameter c.

Schaffer's $\boldsymbol{f}_2$ problem is defined as a two-dimensional vector optimisation. The objective function $\boldsymbol{f}_2(x)$, a vector, has predefined constants:

$$\boldsymbol{f}_2(x) = \left\{f_{1,2}(x),\, f_{2,2}(x)\right\} = \{F_2(x; 0),\, F_2(x; 2)\}$$

Here $\boldsymbol{f}_2(x)$ is a function of a scalar variable x, in general is the objective function a function of a vector variable $\boldsymbol{x}$.

The corresponding optimisation problem aims at simultaneously optimising $f_{1,2}(x)$ and $f_{2,2}(x)$. Obviously multiple solutions coexist, in fact the solution of the optimisation problem is a vector. This vector contains the set of Pareto-optimal points which are of particular interest. Pareto-optimal points may be *inferior* to an other point in the set, but never in all aspects; see also Equation (4.1) on Page 61. Schaffer's bi-objective optimisation problem is, thus, given as:

$$\min_{x \in \mathbb{R}} \left(\boldsymbol{y} = \boldsymbol{f}_2(x)\right) \tag{4.11}$$

The non-inferior solution to this problem complies with the following definition. A decision $x^\star$ is said to be a Pareto solution – if and only if – there does **not** exists another $\bar{x}$ such that $f_{j,2}(\bar{x}) \leq f_{j,2}(x^\star)$, for $j \in \{1, 2\}$, with strict inequality for at least one j. In case of Schaffer's bi-objective optimisation problem, Equation (4.11), the set of Pareto points can be derived analytically. A parametric plot of Schaffer's $\boldsymbol{f}_2(x)$ objective function is given in Figure 4.5 with the Pareto set highlighted. Below we derive the generating function of the Pareto-optimum arc.

For the remainder of this section we suppose processes X^a and X^b have to solve a bi-objective optimisation problem comparable to Schaffer's optimisation problem of Equation (4.11). Equation (4.12) defines the optimisation problems of X^a and X^b which are solved for $x = x_a$ and $x = x_b$ respectively. Their offered operation spaces consist of tuples $\langle f_{a:1},\, f_{a:2}\rangle$ and $\langle f_{b:1},\, f_{b:2}\rangle$. Let $f_1(x)$ and $f_2(x)$ be parameterised versions of $F_2(x; c)$ with $c = \alpha_1$ and $c = \alpha_2$ respectively. As an example, let $f_1(x)$ model the resource utilisation and $f_2(x)$ model the corresponding distortion. The processes X^a and X^b, both have to set an internal parameter x which selects an appropriate combination of the resource utilisation and the distortion. One can think of x as being the number of iterations in an iterative signal processing algorithm. The vector optimisation problem that X^a and X^b face is thus defined by:

$$\min_x \boldsymbol{f}(x) = \min_x \{f_1(x),\, f_2(x)\} = \min_x \{F_2(x; \alpha_1),\, F_2(x; \alpha_2)\} \tag{4.12}$$

We apply the ϵ-constraint method [Haimes et al., 1989], which transforms the multiobjective optimisation problem in a constrained nonlinear optimisation problem. Subsequently, this nonlinear optimisation problem is solved by forming its Lagrange function and checking the Karush-Kuhn-Tucker (KKT) necessary conditions [see, e.g.,

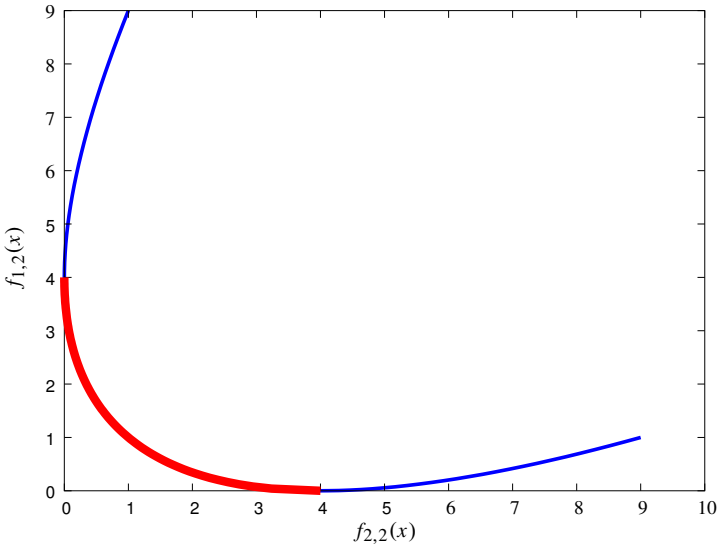


Figure 4.5: Parametric plot of Schaffer's objective function for $-1 \leq x \leq 3$. The Pareto-optimum solution is the highlighted lower left arc of the curve.

Bertsekas, 1999, Prop. 3.3.1]. Without loss of generality, we limit our evaluations to the case that α_2 exceeds α_1 ($\alpha_1 \leq \alpha_2$), which circumvents tedious checking of conditional clauses.

The ϵ-constraint method regards the objective *vector* function of Equation (4.12) and selects one entry of the vector to be its principal objective function. All other entries are turned into parameterised constraints of the optimisation problem. So, the ϵ-constraint method effectively transforms Equation (4.12) into Equation (4.13).

$$\begin{aligned} &\min_x f_1(x) \\ &\text{subject to, } f_2(x) \leq \epsilon_2 \end{aligned} \tag{4.13}$$

The parameter ϵ_2 gets a value larger than the so-called *utopia point* of $f_2(x)$, which is the unconstrained minimum of $f_2(x)$. Let $f_2(\bar{x}) = \min_x f_2(x)$ be the utopia point of $f_2(x)$, then ϵ_2 is defined as $\epsilon_2 = f_2(\bar{x}) + \bar{\epsilon}_2$ for some $\bar{\epsilon}_2 > 0$. In case of Equation (4.12) we have $f(\bar{x})_2 = 0$ for $\bar{x} = \alpha_2$. Note that $\epsilon_2 > f_2(\bar{x})$. Generating the set of non-inferior solutions of the constrained optimisation problem generally requires multiple evaluations of Equation (4.13). For each evaluation ϵ_2 is set to an appropriate value, which makes the constraint *active*, i.e., $(f_2(x) = \epsilon_2)$. A parametric solution with respect to ϵ_2 is referred to as the tradeoff function. Although a parametric solution is preferred it is generally not possible to derive one. In this particular case, however, we can derive the tradeoff function for Equation (4.13).

Equation (4.13) is solved using the Lagrange multiplier method. The Lagrange

function [see e.g. Bertsekas, 1999, Sec. 3.1.3] of Equation (4.13) is given as:

$$L = f_1(x) + \lambda\,(f_2(x) - \epsilon_2) \tag{4.14}$$

in which the generalised Lagrange multiplier ($\lambda \geq 0$) controls the tradeoff between $f_1(x)$ and $f_2(x)$. The optimum of Equation (4.14) corresponds with the optimum of Equation (4.13). Setting $\lambda = \infty$ effectively ignores $f_1(x)$ and the optimum value of x corresponds to the $\{x|f_2(x) = \epsilon_2, \min f_1(x)\}$. At the other extreme of the domain of λ we have $\lambda = 0$. Here $f_2(x)$ is ignored and consequently the optimum value yields the utopia point of $f_1(x)$. However because of the interdependence of $f_1(x)$ and $f_2(x)$, the utopia points of $f_1(x)$ and $f_2(x)$ are *infeasible*. The utopia point of $f_2(x)$ is disqualified as a solution due to the definition of ϵ_2. In order to disqualify the utopia point of $f_1(x)$ we restrict the domain of λ to $\lambda > 0$.

The Karush-Kuhn-Tucker (KKT) necessary conditions follow by taking the partial derivative of Equation (4.14) with respect to x and λ. Collecting the KKT necessary conditions and above derived conditions we have:

$$\frac{\partial L}{\partial x} = 0; \qquad 2\,(x - \alpha_1) + 2\,\lambda\,(x - \alpha_2) = 0 \tag{4.15a}$$

$$\frac{\partial L}{\partial \lambda} = 0; \qquad (x - \alpha_2)^2 = \epsilon_2 \tag{4.15b}$$

$$\lambda > 0; \tag{4.15c}$$

When we solve Equation (4.15a) for λ we find

$$\lambda = -\frac{x - \alpha_1}{x - \alpha_2} \tag{4.16}$$

Equation (4.15b) makes the inequality constraint $f_2(x) \leq \epsilon_2$ an *active* one: $f_2(x) = \epsilon_2$. The domain of x that yields a non-inferior solution follows from the combination of Equations (4.16) and (4.15c).

$$\alpha_1 < x < \alpha_2 \tag{4.17}$$

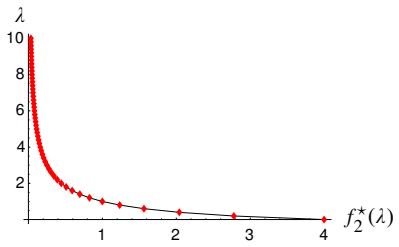
The tradeoff function, which yields all possible non-inferior solutions, is defined in a few steps. First select a valid value for the tradeoff parameter λ, solve $x = x^\star$ from Equation (4.16), and finally substitute $x^\star$ in the objective function $\boldsymbol{f}(x)$.

The optimum value $x = x^\star$, given the tradeoff λ, follows from solving $x^\star$ out of Equation (4.16);

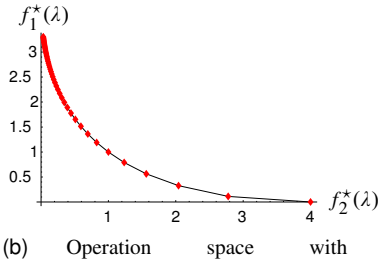
$$x^\star = \frac{\lambda\,\alpha_2 + \alpha_1}{\lambda + 1} \tag{4.18}$$

Consequently, substitution of Equation (4.18) in Equation (4.12), with $\lambda > 0$, yields

$$\begin{aligned} f_2(x^\star) \triangleq f_2^\star(\lambda) &= \left(\frac{\alpha_1 - \alpha_2}{\lambda + 1}\right)^2 \\ f_1(x^\star) \triangleq f_1^\star(\lambda) &= \lambda^2 \left(\frac{\alpha_1 - \alpha_2}{\lambda + 1}\right)^2 \\ &= \lambda^2\, f_2^\star(\lambda) \end{aligned} \tag{4.19}$$



(a) Tradeoff space with $\alpha_1 = -1, \alpha_2 = 1$.



(b) Operation space with $\alpha_1 = -1, \alpha_2 = 1$.

Figure 4.6: Design space.

Figure 4.6(a) plots λ as a function of $f_2^\star(\lambda)$, showing the tradeoff for distortion. Figure 4.6(b) plots the operation space of a process X with the same fixed parameters, $\alpha_1 = -1$ and $\alpha_2 = 1$. The operation space corresponds to a plot of $f_1^\star(\lambda)$ as a function of $f_2^\star(\lambda)$. Observe that the operation space is not uniformly distributed since $f_2^\star(\lambda)$ is non linear in λ (Figure 4.6(a)).

The optimal values of the objectives Equation (4.19) have distinct limits, which are readily derived:

$$\lim_{\lambda \downarrow 0} \begin{cases} f_1^\star(\lambda) & = 0 \\ f_2^\star(\lambda) & = (\alpha_1 - \alpha_2)^2 \end{cases} \qquad \lim_{\lambda \to \infty} \begin{cases} f_1^\star(\lambda) & = (\alpha_1 - \alpha_2)^2 \\ f_2^\star(\lambda) & = 0 \end{cases} \tag{4.20}$$

As a final rewriting we derive the generating function of the Pareto-optimum solution of Schaffer's $\boldsymbol{f}_2$ problem, which has been plotted in Figure 4.5. To that extent, let $f_2^\star(\lambda) \triangleq \xi$, which implies:

$$\lambda = \frac{\alpha_1 - \alpha_2}{\xi} \sqrt{\xi} - 1$$

Consequent substitution in Equation (4.12), for $0 < \xi \leq (\alpha_1 - \alpha_2)^2$, yields:

$$\begin{aligned} f_2^\star(\lambda) &= \xi \\ f_1^\star(\lambda) &= \left(\frac{\alpha_1 - \alpha_2}{\xi} \sqrt{\xi} - 1 \right)^2 \xi \\ &= \xi - 2\,(\alpha_1 - \alpha_2)\sqrt{\xi} + (\alpha_1 - \alpha_2)^2 \\ &= \left(\sqrt{\xi} - (\alpha_1 - \alpha_2) \right)^2 \end{aligned} \tag{4.21}$$

4.4.2 Nonlinear optimisation

Process Z has to optimally combine instantiations (modes of operation) of processes X^a and X^b. In order to mathematically construct this optimisation problem, let $f_{i:1}$ ($i \in \{a, b\}$) represent the resource utilisation and $f_{i:2}$ represent the distortion of an object as offered by process X^i ($i \in \{a, b\}$). The offered operations spaces of X^i, $\langle f_{i:1}, f_{i:1} \rangle$ have been derived in the previous section.

To illustrate the evaluation of underlying optimisation problem, let us assume that process Z minimises the distortion in a balanced way: Z takes the average value of the offered distortions ($f_{a:2}$ and $f_{b:2}$) and increments this value with a penalty, which depends on the difference of the two selected distortion measures. A constant factor, γ say, assigns a relative weight to the significance of combining objects with *comparable* distortion values. The total resource utilisation budget (ρ) of the system constraints the optimisation problem; the sum of resource utilisation value of the selected operation points should not exceed ρ: $f_{a:1} + f_{b:2} \leq \rho$. In the remainder of this section, we solve the optimisation problem parametrically, for a range of ρ.

The above informal description yields the following mathematical formulation of the (non-linear) optimisation problem that Z faces.

$$\begin{aligned} &\min_{\boldsymbol{f}} \; h(\boldsymbol{f}) = \frac{1}{2}(f_{a:2} + f_{b:2}) + \gamma \; (f_{a:2} - f_{b:2})^2 \\ &\text{subject to, } \; f_{a:1} + f_{b:1} \leq \rho \end{aligned} \tag{4.22}$$

We solve Equation (4.22) by applying the Lagrange multiplier theory [Bertsekas, 1999]. Substitution of Equation (4.19) in Equation (4.22) yields an ordinary nonlinear constraint optimisation problem.

$$\begin{aligned} &\min_{\lambda_a, \lambda_b} \frac{1}{2} \left(f^{\star}_{2,a}(\lambda_a) + f^{\star}_{2,b}(\lambda_b)\right) + \gamma \; \left(f^{\star}_{2,a}(\lambda_a) - f^{\star}_{2,b}(\lambda_b)\right)^2 \\ &\text{subject to, } \; f^{\star}_{1,a}(\lambda_a) + f^{\star}_{1,b}(\lambda_b) \leq \rho \end{aligned} \tag{4.23}$$

The operation spaces of $\{f^{\star}_{1,i}(\lambda_i), f^{\star}_{1,i}(\lambda_i)\}$ $(i \in \{a, b\})$ depend on the difference of α_2 and α_1 of each of the processes X^a and X^b, see Equation (4.19). Let $\sqrt{\sigma_i} = (\alpha_{2,i} - \alpha_{1,i})$ $(i \in \{a, b\})$. Since $\alpha_1 \leq \alpha_2$ we have $\sqrt{\sigma_i} \geq 0$. Without loss of generality we furthermore assume $\sigma_a \leq \sigma_b$. We evaluate Equation (4.23) parametrically in ρ. In connection with Equation (4.20), the feasible interval for ρ is defined as $0 < \rho < \sigma_a + \sigma_b$.

The Lagrange function of Equation (4.23) is given by Equation (4.24), in which the functions $\{f^{\star}_{1,i}(\lambda_i), f^{\star}_{2,i}(\lambda_i)\}$ $(i \in \{a, b\})$ have been substituted from Equation (4.19). Here, μ is the Lagrange multiplier, which effectuates the constraints of Equation (4.23).

$$\begin{aligned} L = \frac{1}{2}\left(\frac{\sigma_a}{(1+\lambda_a)^2} + \frac{\sigma_b}{(1+\lambda_b)^2}\right) + \gamma \left(\frac{\sigma_a}{(1+\lambda_a)^2} - \frac{\sigma_b}{(1+\lambda_b)^2}\right)^2 + \\ \mu \left(\frac{\sigma_a \, {\lambda_a}^2}{(1+\lambda_a)^2} + \frac{\sigma_b \, {\lambda_b}^2}{(1+\lambda_b)^2} - \rho\right) \end{aligned} \tag{4.24}$$

The KKT necessary conditions for Equation (4.24) are

$$\frac{\partial L}{\partial \lambda_a} = 0; \qquad \mu \left(\frac{-2\,\sigma_a\,{\lambda_a}^2}{(1+\lambda_a)^3} + \frac{2\,\sigma_a\,\lambda_a}{(1+\lambda_a)^2} \right) - \left(\frac{\sigma_a}{(1+\lambda_a)^3} \right) - \frac{4\,\sigma_a \left(\frac{\sigma_a}{(1+\lambda_a)^2} - \frac{\sigma_b}{(1+\lambda_b)^2} \right) \gamma}{(1+\lambda_a)^3} = 0 \tag{4.25a}$$

$$\frac{\partial L}{\partial \lambda_b} = 0; \qquad \mu \left(\frac{-2\,\sigma_b\,{\lambda_b}^2}{(1+\lambda_b)^3} + \frac{2\,\sigma_b\,\lambda_b}{(1+\lambda_b)^2} \right) - \left(\frac{\sigma_b}{(1+\lambda_b)^3} \right) + \frac{4\,\sigma_b \left(\frac{\sigma_a}{(1+\lambda_a)^2} - \frac{\sigma_b}{(1+\lambda_b)^2} \right) \gamma}{(1+\lambda_b)^3} = 0 \tag{4.25b}$$

$$\frac{\partial L}{\partial \mu} = 0; \qquad \frac{\sigma_a\,{\lambda_a}^2}{(1+\lambda_a)^2} + \frac{\sigma_b\,{\lambda_b}^2}{(1+\lambda_b)^2} = \rho \tag{4.25c}$$

$$\mu \geq 0; \tag{4.25d}$$

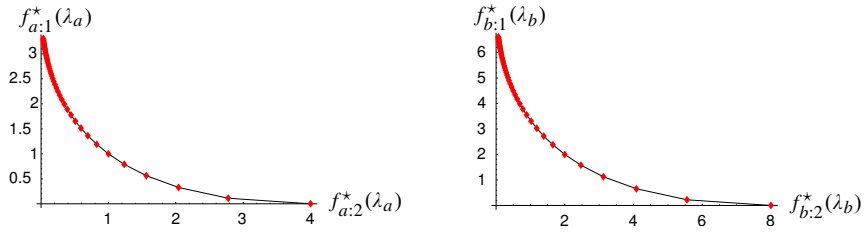
Solving this scheme of expressions requires some structuring. Note that previously we solved Equation (4.19) for the case that λ_a and λ_b are real-valued and non-negative. Solving Equation (4.25c) for λ_a yields Equation (4.26).

$$\lambda_a = \frac{-\left(\rho\,(1+\lambda_b)^2 - {\lambda_b}^2\,\sigma_b\right) \pm (1+\lambda_b)\,\sqrt{\sigma_a}\,\sqrt{\rho\,(1+\lambda_b)^2 - {\lambda_b}^2\,\sigma_b}}{\left(\rho\,(1+\lambda_b)^2 - {\lambda_b}^2\,\sigma_b\right) - (1+\lambda_b)^2\,\sigma_a} \tag{4.26}$$

With some tedious formula manupilations, it is possible to bind the solution, however it is hard to derive at a pure analytical solution. Numerical evaluation on the other hand, e.g. applying Newton's method or the secant method, is straightforward. Consider, for illustration purposes, an example with fixed parameters. Let $\gamma = 1$, $\sqrt{\sigma_a} = 2$, and $\sqrt{\sigma_b} = 2\sqrt{2}$. The operation space of X^a and X^b are respectively given in Figures 4.7(a) and 4.7(b). Combining objects from these operation spaces while minimising the overall, balanced, distortion within a given budget (ρ), results in the operation space of Z. The operation space is illustrated in Figure 4.8(a). The corresponding qualitative objective of Z is given in Figure 4.8(b). Figure 4.8(a) plots the first part of Equation (4.23), the average value of the respective distortions. Figure 4.8(b) plots the second part of Equation (4.23), the quadratic difference between the respective distortions.

4.5 Conclusion

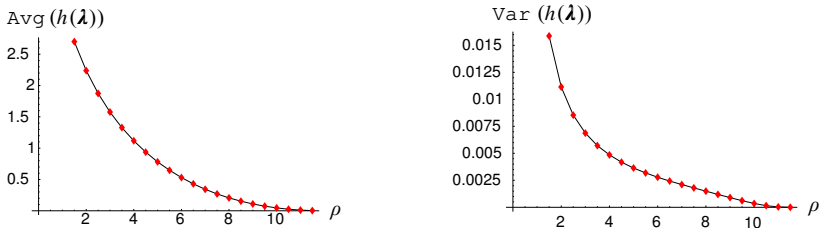
In this section, we have demonstrated the generation and usage of operation spaces in homogeneous merger type of networks (Section 3.4.2). In this simple case we managed to handle all but the system wide operation spaces analytically. Now already some tedious formula manipulation was required so as to arrive at analytical expressions operation spaces. In practical situations significantly more expert knowledge is required to keep operations spaces analytically tractable. In many practical situations



(a) X^a: with $\sqrt{\sigma_a} = 2$.

(b) X^b: with $\sqrt{\sigma_b} = 2\sqrt{2}$.

Figure 4.7: Operation spaces of X^a and X^b.



(a) Average distortion of operation points of Z.

(b) Difference in distortion of constituent objects of operation points of Z.

Figure 4.8: Design space of Z.

components are often prohibitively complex and analytical solutions can not be generated. Components that use many subcomponents are notorious in this respect; the evaluation of Z, above, is more tedious than the evaluation of X. An all analytical description of a system is generally not feasible, but approaches that rely on emulation or tabulation can be applied successfully, see Section 6.1.3. What is important is the monotonic nature of the operation space. The Pareto curve of Figure 4.5 is typical in this respect.

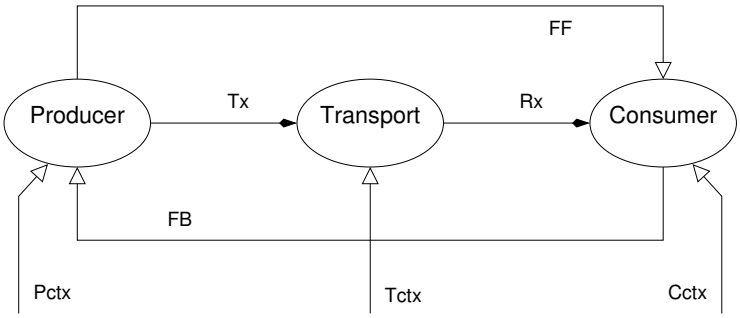
Five

Context-aware process networks

DEMOCRATIC processing is a direct consequence of the reflections in the preceding chapters on coordination in communicating systems. We regard a communicating system as a *composition* of *autonomously* behaving entities (subsystems). The democratic aspect emphasises the distributive character of the required coordination among system entities. Distributed coordination assists in managing the complexity of a system, and facilitates the necessary flexibility to handle subsystems that are in a constant state of flux. Distributed coordination requires operational information exchange. In Chapter 3 we presented ARC, a framework for quality of service (QoS) negotiations based on abstraction, adaptation, and cooperation. ARC effectively implements exchange of non-functional or operational information; in view of Chapter 2, ARC combines constructs from an engineering, a computational, and an informational view. Adaptation facilitates flexible autonomous entities with necessary context awareness, whereas cooperation involves sharing of context with neighbouring entities.

In this chapter we concentrate on stream-based processing structures of context-aware entities; a general dataflow network with occasional coordination (control) events. An implementation of such a network requires the same concepts as the ones ARC is based on: abstraction, adaptation, and cooperation. In this chapter we take a formal approach to coordination, which allows an evaluation of the results. We will especially target compositionality here.

A system is a composition of autonomous entities, therefore, the exchange of operational information should be *decoupled* from the mainstream dataflow: asynchronous coordination. Entities must have the opportunity to decide autonomously whether or not to adapt to a possible change of context of its milieu. Likewise, entities must have the opportunity to decide autonomously whether or not to expose a possible change of context to its milieu. Figure 5.1 exemplifies a common producer-consumer network with in between a transport network. The links Tx and Rx (solid arrows) carry the mainstream dataflow. All components are context aware; they (may) adapt to their respective context change events, Pctx, Tctx and Cctx (open arrows) respectively. The producer and consumer in the diagram are cooperative; they expose and incorporate operational in-



FB: feedback, FF: feedforward, and Xctx: context sensing.

Figure 5.1: Context-aware producer-consumer system.

formation in a feedforward and feedback manner. As an example, the producer may inform the consumer about the future workload (FF), which allows the consumer to select an appropriate (efficient) mode of operation. Vice versa, the consumer may inform the producer about the current state of the network and display capabilities (FB), which allows the producer to select an appropriate mode of representation. The figure demonstrates the relevance of asynchronous coordination in complex communicating systems. Coupling of the pace at which operational information is exchanged between a producer and a consumer can be awkward because they typically have their own pace at which context changes can be effectively incorporated. Imposing events at other times may even be counterproductive. In addition, synchronous coordination imposes strict constraints on the behaviour of the transport network between the producer and consumer. In complex situations it might be difficult or even impossible to guarantee these constraints. Examples that come to mind are networks that suffer from packet loss or out-of-order delivery.

Unfortunately, context awareness comes at a price. Since the behaviour of an entity depends on its context, the entity becomes *indeterminate*. Moreover, a composition of indeterminate entities yields a non-deterministic system. In this chapter, nonetheless, we develop a model of computation that supports democratic processing. Our context-aware process network (CAPN) model of computation has necessarily indeterminate constructs, yet compositionality is retained through the isolation of context dependence. We show that the modelling of context-dependent behaviour is closely related to the modelling of *fairness*.

The functional behaviour of a system of communicating entities can be unambiguously defined through a so-called model of computation (MoC). Many of these MoCs have been developed, each serving a different purpose. Whether or not a specific MoC meets the "unambiguous" property was briefly addressed in Section 2.1.3. The association of a MoC with a performance model uniquely relates non-functional information with functional behaviour and so it facilitates high-level design methods. A noteworthy approach is the work of Kienhuis and Lieverse et al. In [Kienhuis, 1999] a methodo-

logy is developed for exploring the design space of stream-based functions and their implementation. In [Lieverse et al., 2001] this idea is extended by linking the MoC to a model of architectures and the introduction of a cycle-accurate simulations. The methodology models an application functionally using a stream-based processing model of computation: the Kahn process networks (KPN) [Kahn, 1974]. Evaluation of the Kahn process network yields a *trace* of (discrete) events that, when implemented (mapped) on an appropriate platform, yields performance metrics (time, power, area, etc.) for a given scenario. The KPN was chosen as the MoC to model applications since KPNs are unambiguous and they map naturally to the range of target applications (in the video processing domain).

A Kahn process network is a restricted process network in which autonomous sequential processes communicate peer-to-peer through unidirectional unbounded FIFO buffers. A processes can access a buffer only through a destructive, and potentially blocking, read or through a non-destructive write operation. Using a restricted model of computation is advantageous for the integrity of the system. The prominent aspect of Kahn process networks [Kahn, 1974; Kahn and MacQueens, 1977] is that they provide *determinate* composition of autonomous processes; irrespective of any execution order (schedule) of the processes and irrespective of the underlying memory model, the resulting trace is solely determined by the initialisation of the network. Because the Kahn process network model is determinate it is also *compositional*: two subsystems of a network with equivalent input-output relations can be exchanged without affecting the input-output relations of the encompassing network. In Section 5.2 we show that compositionality cannot be determined from the input-output relations of a component alone.

Obviously, the Kahn process network MoC is unable to naturally model the exchange of those occasional events that propagate operational information, since these events are indeterminate by definition. A commonly applied, but generally unsatisfactory solution is to explicitly model the exchange of operational information. As an example consider modelling the interaction with the user, who by means of pressing a button can influence the course of the processing. Explicit modelling of the user interaction requires the generation of a stream of "on" and "off" pressing events, where every possible sequence of "on" and "off" events represents a scenario. Exhaustive simulation of all possible user interaction scenarios, however unnecessary these may be, is prohibitively time consuming. But the relation between a scenario and a sequence of "on" and "off" events is not clear a priori. Explicit modelling of the context (e.g. the user interaction) fixes the scenario in which the system operates. Although explicit modelling retains the determinate property of the model it sacrifices context awareness.

In short, Kahn process network semantics include asynchronous communication but do not include asynchronous coordination. We chose to extend Kahn's semantics to facilitate asynchronous coordination, which results in a model of computation named *context-aware process networks* (CAPN). The advantage of our approach over alternative MoCs is that CAPN retains the analytical strength of KPN. Moreover, we emphasise the stream-based character of the systems under consideration, yet make (sub)systems truly context aware. Asynchronous coordination allows subsystems to adapt to changes in their milieu and to cooperate with subsystems in their milieu. Because of the asynchronous aspects, the coordination process is also non-intrusive.

Alternative approaches for extending Kahn's semantics exist. CSP, Petri nets, and discrete event systems are noteworthy examples of more generic models of computations. Hoare introduced communicating sequential processes (CSP) [see e.g. Hoare, 1985; Roscoe, 1997] and Peterson introduced Petri Nets [see e.g. Peterson, 1981; Holloway et al., 1997; David and Alla, 2001]. Interesting in this respect is the subset of nets that includes context: contextual nets [see e.g. Montanari and Rossi, 1995; Winkowski, 1998]. Petri Nets might be regarded as an analytical branch of the vast field of discrete event systems [see e.g. Cassandras and Lafortune, 1999].

The characteristics of various models of computations, their strengths and weaknesses, have been studied in [Edwards et al., 1997], who introduced the tagged-signal model. This model acts as a framework for comparing different models of computation. In this model a signal is represented by a string of events (value/tag tuple). The ordering of events in a signal, the relative ordering of signals, and the input-output relations of components (nodes) are different for the various models of computation.

CSP is a model of computation based on rendez-vous communication with well-defined semantics, usually presented in the form of a denotational semantics. The CSP specification allows, under strict conditions, for the decision on network properties such as livelock and deadlock. Petri nets are also often presented denotationally. Petri nets have been studied extensively, which has led to a vast amount of theoretical results. So algorithms exist to decide on significant properties such as deadlock and livelock of Petri nets in finite time and finite memory. Petri nets are often applied for evaluation of control networks. Discrete event systems usually have operational semantics and rely for their evaluation on extensive simulations. Discrete event systems are suitable for specifying distributed systems. Note, however, that simulating large and complex discrete event systems is not a trivial task; a distributed implementation of a discrete event simulation, for instance, requires formally verified methods in order to maintain a coherent notion of *time*.

The reason we did not pursue any of the aforementioned alternatives is the natural fit of Kahn process network MoC and the process structures encountered in communicating systems. Recall the general sensor analysis synthesis actor (SASA) network (Figure 1.1 on Page 2). The figure has been redrawn in Figure 5.2. The network is, except for the cooperative (control) flow c, a textbook example of a dataflow network; from input u to output y.

Evidently, deviations from the Kahn semantics will break the model's determinate property. Consider the ordinary feedback structure of Figure 5.3. Closing the switch `S` induces a change of context. Initially, with `S` open, the process `P` will generate an event on its output stream y for every incoming event on stream u. With `S` closed, the input stream u is effectively *merged* with the output stream y. After receiving an event on u, process `P` may generate any number of events on y; the exact amount of events depends on the implementation of the merge operation. The *indeterminate* behaviour of the merge operation causes these models to lose their compositional property, as was first shown by Brock and Ackermann [Brock and Ackermann, 1981]. This type of indeterminate behaviour is known in literature as the Brock and Ackermann anomaly or *merge* anomaly. Feedback or feedforward structures are generally used to initiate cooperation. In a dataflow network, cooperation regulates *when* adaptations take place. Although Kahn process networks specify perfectly *where* a monitor of any context

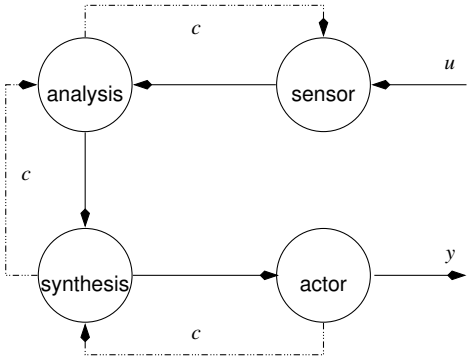


Figure 5.2: SASA, dataflow network.

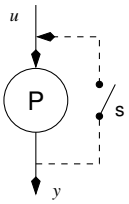


Figure 5.3: Context switch with feedback structure.

parameter resides and *how* context updates can be incorporated in the dataflow, the *when* question is left unanswered. Note that we encountered similar concerns when discussing negotiated QoS in Section 3.3.2, [see West and Schwan, 2001]. The Brock and Ackermann anomaly in a feedback structure reduces to a *causality* like anomaly: the ordering of events at the output of the merge operation depends on the (causal) arrival of events at inputs of the merge operation. The model thus retains its compositional property when augmented with a proper *time* characterisation, which specifies the relative ordering (the when) of arriving feedback events. If possible we can capture the determinate process of ordering events in a so-called *oracle* [Russell, 1990; Panangaden, 1995], which effectively removes the anomaly.

In this chapter we propose to extend the Kahn process network MoC with a basal modelling construct: a register link (REG) with destructive write behaviour. Our implementation-wise trivial extension has far-reaching consequences on the analysis of the resulting process network since it is no longer determinate. However, an analytical evaluation of our extension shows that it is possible to isolate the resulting indeterminate behaviour. Modelling of a REG link yields a parameterised Kahn process network, albeit a varying one, and thus it retains its compositional property. The initial indeterminate behaviour of the REG link is now concentrated into a control stream, of which the properties either impose constraints on the context of the system for proper operation or abstract the context conditions for the system to operate in.

Not surprisingly other researchers, too, chose to implement a register (REG) in their development environment; a register construct maps one-on-one with a hardware implementation and enlarges the scope of a MoC significantly. Basically there exist two distinct concepts of modelling. One either develops a MoC based on discrete-event processing or a MoC based on stream-based processing.

In [de Kock et al., 2000], De Kock et al. propose a select (merge) construct to model infrequent events in a stream-based MoC. A user who presses a button is an example of such an infrequent event. The select construct is incorporated with the previously mentioned design space exploration methodology of [Lieverse et al., 2001]. Lieverse et al. use an executable model in which there is a strict separation between the ap-

plication domain (stream-based KPN) and the implementation domain (discrete events). The methodology relies on extensive simulations, but because of the separation the simulations of the application domain need not be exhaustive. However, when the methodology is extended with a select construct in the application domain, the separation is no longer strict as the select construct, which resides in the application domain, requires (causal) input from the discrete event domain. Consequently, the methodology degrades to a method of exhaustive simulations. As a marginal note, the *fairness* of the select construct complicates matters even further. A different choice of fairness yields a different application behaviour. CAPN, as developed in the chapter, isolates the indeterminate behaviour of a select construct, which in specific cases circumvents the need for exhaustive simulations to analyse a system.

The SPI (System Property Intervals) workbench [Ziegenbein et al., 2002] evaluates possible schedules for heterogeneous systems. SPI implements a non-executable model. The modelling language, much like Kahn semantics, is a coordination language among components that are allowed to apply different MoCs. Components communicate with each other through (destructive read) FIFO links or (destructive write) registers. SPI establishes the bounds on the behaviour of the entire system. To do so, SPI requires an extensive abstraction of the communication, execution pattern, and resource utilisation of each component. It uses an explicit communication model, i.e., the availability of data is a predicate to enable process execution. Data dependent control is modelled stochastically with parameterised uncertainty intervals; this support for handling ranges of parameter values rather than fixed values extends the applicability of the methodology significantly. Unlike SPI, our approach (CAPN) does not require explicit communication and execution patterns. Data dependent control, if any, is hidden in the respective nodes of the network. The lack of an explicit account of communication and execution patterns corresponds to our prerequisite of flexibility as a design parameter (see problem description on Page 4); explicit communication and execution patterns may hinder evolution.

FUNSTATE [Strehl et al., 2001] is an internal data representation developed in parallel with SPI. FUNSTATE distinctly separates control and data, whereas SPI – like CAPN– does not enforce this separation. FUNSTATE enables methods for formal verification (deadlock free, no overload, etc.) and methods for design and evaluation of schedules. Like SPI, FUNSTATE has limited support for asynchronous coordination. FUNSTATE is a discrete-event-based MoC related to coloured Petri Nets. For the formal analysis of CAPN we make use of the so-called synchronous data flow model. Other researchers have proposed an extension to this model in order to support (parameterised) indeterminate behaviour. Noteworthy examples of parameterised dataflow and extensions of dataflow with non-functional properties can be found in [Bhattacharya and Bhattacharyya, 2001] and [Park et al., 2002].

In the remainder of this chapter we first introduce Kahn process networks and the related concept of dataflow networks. We then introduce non-determinism and present the register extension with applications and analysis. We conclude with an example to demonstrate asynchronous coordination by means of the register construct. As a matter of fact, our context-aware process network (CAPN) MoC is a formal implementation of the ARC framework of Chapter 3 for stream-based networks with occasional context events.

5.1 Kahn process networks

Kahn process networks are deterministic. Their functional behaviour is independent of the applied execution order of the processes in the network, provided that the implemented schedule is a feasible one. In general, Kahn process networks lack a global schedule and do not impose a specific memory model. Because of their deterministic property, Kahn process networks are frequently applied to model, evaluate, and design high-performance digital signal processing systems [Edwards et al., 1997; Lieverse et al., 2001; Deprettere et al., 2002; Stefanov et al., 2002].

The family of dataflow models of computations as presented in [Lee and Parks, 1995; Bhattacharyya et al., 1999; Najjar et al., 1999], are a subset of the Kahn process network (KPN) model of computation. Their discriminating difference is that dataflow models do, and the Kahn model does not have a global schedule. In KPN, process nodes are autonomous processes with explicit communication semantics. In a dataflow MoC the semantics of the process nodes are restricted further by introducing an explicit or implicit *trigger* signal. Processes synchronise on, possibly parameterised, trigger events. With restricted semantics enforced, the resulting network has reduced complexity, which means that more a priori and more detailed analysis is possible.

The dataflow model of computation with the most restricted semantics is the so-called synchronous dataflow (SDF) model. In the SDF model each process has static behaviour, which means that on each invocation of a trigger event (the *firing*) the number of processed tokens is statically specified. Exploiting the static behaviour of SDF allows for an analytical evaluation of the network. The Ptolemy system [Lee, 2001] uses SDF to design embedded real-time systems with guaranteed deadlock and livelock behaviour. Parks presents a method to derive global schedules that bound (minimise) the memory usage of an SDF network [Parks, 1995]. SDF uses implicit trigger events; the firing of a process implies the availability of a trigger event without interpreting the actual content of that event.

The cyclo-static dataflow (CSDF) model yields a useful extension of the SDF model for which practical analysis techniques are available [Bilsen et al., 1996]. In CSDF the nodes implement a finite set of operation modes, e.g, $\{f_0, \cdots, f_{N-1}\}$, where f_i corresponds with the operation mode of an SDF node. The modes of operations are sequentially and repeatedly executed; the i-th invocation triggers the execution of operation $f_{i \bmod N}$. Consequently the, statically determined, number of tokens that is processed per firing is a cyclic-extended sequence because the number of tokens that is processed at the i-th invocation is determined by mode of operation f_i.

A logical extension of SDF and CSDF is the so-called dynamic dataflow (DDF) MoC. Its basic implementation, the Boolean dataflow (BDF), has processes with two distinct modes of operation, each mode corresponding to an SDF process. On every invocation of a process in BDF, a trigger (control) event is received with a “true” or “false” value. The content of the event selects the mode of operation for the current firing. Buck shows that a global schedule can only be derived for the BDF MoC for a specific class of problems [Buck, 1993].

The Kahn process network MoC does not restrict the communication synchronisation. Lieverse et al. exploits this property to evaluate possible implementations of video processing filters [Lieverse et al., 2001]. The application is functionally specified

in the KPN model of computation, which results in a large set of possible schedules. Subsequently the set of schedules is associated with an architecture model, which allows one to emulate the consequences of a schedule on an aspect of resource utilisation, such as time, memory, etc. Systematically varying the parameters of the architecture and the application models results in a methodology for design space exploration.

5.1.1 KPN semantics

Operational semantics

The *operational* semantics of Kahn process networks is as follows. Kahn process networks connect *nodes* that habitat an autonomous sequential *process*. A node has *ports*: input ports and output ports. Ports of distinct nodes are connected through *channels*. An output port connects to at most one input port, and similarly an input port connects to at most one output port. The channel between an output port and an input port is a unidirectional unbounded FIFO (first-in first-out) queue. Processes are allowed to access their ports with two possible operations[1]: put(·) , a non-blocking *write* to an output port, and get(·) , a blocking *read* from an input port. As a consequence, scanning or monitoring of a port is not possible. Both the put(·) and the get(·) operations are determinate – atomic – operations.

Practical implementations of Kahn process networks require the determination of the execution order of the processes in the network: the *schedule*. An intuitive approach to generate a feasible schedule is the *demand driven* schedule of [Kahn and MacQueens, 1977]. In this approach the network is initiated with *empty* channels and if necessary augmented with *source* and *sink* processes. An arbitrary process receives the thread of control and starts execution. Whenever the process tries to read from an empty channel, the process marks the channel as *hungry* and hands over the thread of control to another process. A process that writes to a hungry channel will un-mark the channel as being hungry and also hands in the thread of control. Parks noted that the sketched execution model may still require unbounded memory. In [Parks, 1995], an execution model is presented that guarantees a bounded memory execution if at all possible.

Example 5.1 (Kahn Switch) *Consider the Kahn network of* Figure 5.4*, which shows a switch composed of Kahn processes. The incoming stream u is processed either through P_1 or through P_2. The effective route is controlled by the control (trigger) stream c, which instructs P_0 to route data through x_1 or x_2 and instructs P_3 to retrieve data from x_3 or x_4. The coordination among P_0 and P_3 is fully synchronised; P_0 expects a control event for every event on u. Equivalently, P_3 relies on c to decide whether to retrieve data from P_1 or P_2. Because P_3 is a Kahn process, it will block when reading from an empty channel.*

Kahn process networks with explicit (or implicit) control c are referred as dataflow processes [Lee and Parks, 1995]. Consequently, compositions of these nodes are called dataflow process networks. The processes P_1 and P_2 have implicit control. On every implicit control event, process P_1 (P_2) will take an event from x_1 (x_2) and output the result on x_3 (x_4).

[1] We refer to these operations as put(·) and get(·) following [Kahn and MacQueens, 1977; Ptolemy, 1995–2002]. Other authors use different names, e.g., [Kienhuis, 1999; Lieverse et al., 2001] use read() and write().

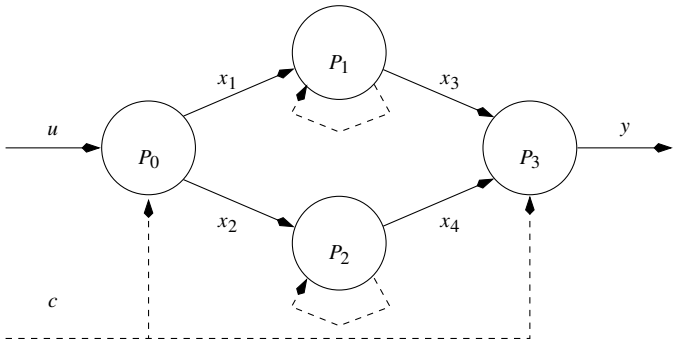


Figure 5.4: Kahn process network: a (dataflow) switch.

The switch of Figure 5.4 *exemplifies a multi modal network. Processing takes either the upper route or the lower route. There are multiple options to generate the control stream c that selects the actual mode of operation. In case of* SDF*, c is a constant stream that reduces the network to either the upper path or the lower path. In a dynamic dataflow MoC c is context dependent. For instance, we may choose to let process P_0 generate the control stream c based on the actual value of u, or alternatively, let process P_1 generate c based on the most recent value of x_3 or x_4. In general is c is generated by an external (coordinating) process.*

Denotational semantics

Kahn processes are determinate and consequently their composition, a Kahn process network, is also determinate. The following equivalent *denotational* semantics [Winskel, 1993] of Kahn processes illustrates this conjecture formally.

A Kahn process implements a *continues* function[2]. Let $u = \langle \lambda_0, \lambda_1, \cdots, \lambda_n \rangle$ denote an input sequence of events to a process P and let y denote the corresponding output sequence. The input sequence has a so-called *partial order* if the elements of u have an ordered index relation: $\lambda_i \leq \lambda_{i+1}$, for $0 \leq i < n$. A prefix order on u is a partial order that partitions u in overlapping subsequences: $\langle u_0, u_1, \cdots, u_k \rangle$, where u_0 equals the empty set ($u_0 = \langle \perp \rangle$) and $u_k = u$. The subsequence u_i is a *prefix* of subsequence u_j, denoted as $u_i \sqsubseteq u_j$, if all elements of u_i are in u_j and those elements not in u_j have a higher index than all elements in u_i. As an example, let $u_i = \langle \lambda_0, \cdots \lambda_i \rangle$ and $u_j = \langle \lambda_0, \cdots, \lambda_i, \lambda_{i+1}, \cdots \lambda_j \rangle$, then if $u_i \sqsubseteq u_j$, we have $\lambda_i \leq \lambda_{i+h}$ for all $h = 1 \cdots j - i$.

A *monotonic* process implements a monotonic function F that transfers a partial order (u) into another partial order (y): $y = F(u)$. A monotonic function F is defined as

$$u_i \sqsubseteq u_j \Longrightarrow F(u_i) \sqsubseteq F(u_j). \tag{5.1}$$

Monotonicity corresponds to the intuition that input events that arrive at the input port after computations have started, cannot influence any current event of the output

[2] See e.g. [Roscoe, 1997, Appendix A.1] for a thorough introduction.

stream. An important practical implication is that processes can start processing as soon as they have sufficient input to generate (part of) their output. A coarse grain example is found in pipelined arithmetic and a fine grain (bit-level) example is found in on-line arithmetic [M.D. Ercegovac, 1989]. The latter example requires a redundant number of representations in order to force monotonicity of arithmetic functions. Models of communication that have a notion of time define the (temporal) causality of events. Monotonicity defines *causality* for models of communication that operate on (ordered) sequences rather than implement global time.

The infinite extension of a partial order $u \rightarrow u^{\infty}$ requires u to have a greatest lower bound $\sqcap u$ and a least upper bound $\sqcup u$. Let u again be a prefix order, then the $\sqcap u$ of u equals the, empty, bottom element $\perp$, which is a prefix for u. Let y be the set of upper bound elements, thus u is prefix for every element of y; $t \in y$ is the least upper bound when t is prefix too of every element of y. The extension $u \rightarrow u^{\infty}$ makes u^{∞} a complete partial order. Subsequently, a continues function F is defined as

$$F(\sqcup u^{\infty}) = \sqcup F(u^{\infty}). \tag{5.2}$$

The definition of a continues function implies monotonicity [Roscoe, 1997]. The reverse, however, is not true; not all monotonic functions are continues functions. "Fortunately this [...] is not a problem in practice since practical monotonic processes are invariably continuous" [Lee and Sangiovanni-Vincentelli, 1998]. This is an important conjecture because it implies that Tarski's theorem[3] applies in practical situations. Tarski's theorem for complete partial orders states that a monotone function F that operates on complete partial orders has a *least fixed point*: $x = F(x) = \sqcup\{y | y \leq f(y)\}$.

The existence of a least fixed point implies that monotone functions are determinate. Monotonicity and continuity therefore are preserved under function composition. A composition of monotone (functional) processes yields again a monotone (functional) process. A constructive proof of Tarski's theorem [Roscoe, 1997, Appendix A.1] is based on the recursive application of the function F, starting from the empty sequence $\perp$. The proof implicitly contains an iterative procedure for finding the least fixed point solution. The procedure closely resembles the operational semantics of the demand driven schedule we sketched on page 86 [Winskel, 1993].

5.1.2 Synchronisation

A communicating system is a natural representative of a parallel program. The proper execution of any parallel program relies on underlying synchronisation mechanisms and their implementation. Kahn semantics implement *asynchronous* communication; there is no (global) notion of time that *synchronises* events. Asynchronous communication has been recognised for long to be an efficient implementation of communication. In [Miller, 1965, chap. 10] a chapter is devoted to "speed independent switching theory" for improving the integrity of a design. More recently, asynchronous designs methods have been applied to save energy [Rabaey, 2001; Wan et al., 2001].

Van Gemund presents a clear analysis of the necessary synchronisation constructs for implementing parallel programs [van Gemund, 1996]. There are three basic syn-

[3] Tarski's theorem is also referred to as the Knaster–Tarski fixed point theorem.

chronisation constructs (Definition 5.1). Practical models of computation implement each of these constructs in one way or another.

Definition 5.1 (Synchronisation constructs [van Gemund, 1996])

i. CONDITION SYNCHRONISATION *(CS); Static synchronisation of precedence relations among autonomous processes.*

ii. MUTUAL EXCLUSION *(MUTEX); Dynamic synchronisation of precedence relations. Mutual exclusion organises atomic access to shared resources.*

iii. CONDITIONAL CONTROL FLOW *(CCF); Data dependent control flow.*

❑

Condition synchronisation (CS) is found when a process P cannot proceed computations before, say, processes S_0 and S_1 have provided it with the right amount of data. Mutual exclusion (MUTEX) is associated with contention of (shared) resources. A MUTEX is a dynamic form of CS. Conditional control flow (CCF) captures the data dependency of a program.

Van Gemund developed the above definition for the purpose of performance modelling [Pamela, 1993–2002]. Careful implementation of the basic synchronisation constructs facilitates the automatic derivation of a parameterised performance model.

The semantics of Kahn include the CS and CCF constructs but lack the MUTEX construct. The MUTEX construct is a prerequisite for asynchronous coordination; note that in contrast to CS and CCF, the MUTEX construct is an indeterminate primitive. To illustrate the advantages of asynchronous coordination, consider a typical synthesis analysis network. Maintaining synchronous coordination among processes is awkward when multiple developers, or worse, multiple disciplines, are involved. In such a case, the development process of the network requires careful coordination; moreover, it is limited with respect to the flexibility of developments. Typically, the analysis process has dedicated points where it is useful to incorporate cooperative information. Likewise, the synthesis process has dedicated points in the process where sensible cooperative information can be provided. These mode-switch points of the synthesis and analysis processes usually do not coincide. Hence a desire for asynchronous coordination.

5.2 Indeterminate processes

Introducing nondeterminacy roughly comes in two flavours. One can dynamically change the topology of a network, or one can apply indeterminate processes in a network. Here, we will pursue the latter. Changing the topology of a network dynamically is an interesting topic though. In [de Bruin and Nienhuys-Cheng, 1998] a linear dynamic extension of the topology is studied. It turns out that this type of extension is deterministic. An example of a linear dynamic extension is the creation of a network that implements the "Sieve of Eratosthenes" [Kahn and MacQueens, 1977; de Bruin and Nienhuys-Cheng, 1998]. The network generates a sequence of prime numbers based on the filtering of multiples. Newly determined prime values define new filter components. Suppose that at some instance the network topology contains filters for the primes 2, 3, 5, 7. Then, when a test sequence of $\langle 8, 9, 10, 11\rangle$ enters the network, 8

will be filtered out by the 2-filter since 2 divides 8. Similarly, 9 will be filtered out by the 3-filter and 10 by the 2-filter again. Since 11 is a prime it will make it all the way through the 7-filter and subsequently a new filter component (the 11-filter) is linearly added to the network.

We concentrate on the type of nondeterminism that results from the application of indeterminate processes. We are particularly interested in indeterminate processes caused by context awareness. Processes that are willing to adapt to changes in their milieu must be able to sense their context. A prominent example of an indeterminate primitive is the *merge* construct. A merge primitive allows an application to react to incoming events; either a changed condition (outage of a wireless link) or a changed objective (invocation of a new application by a user). A merge construct implements the MUTEX construct of Definition 5.1.

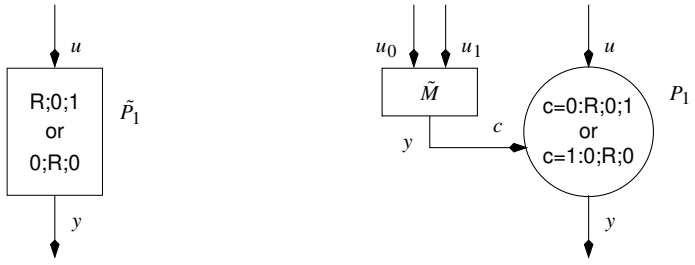
The implementation of a merge construct is fairly straightforward in any executable (operational) model. We conducted experiments, implementing a merge construct, using the Pamela (C) run-time library [Pamela, 1993–2002] and the Ptolemy (Java) suite [Ptolemy, 1995–2002]. Additional examples can be found in literature. For example, we mentioned the register construct implemented in the SPI workbench [Ziegenbein et al., 2002] and the introduction of a *select* construct in YAPI [de Kock et al., 2000] a programming interface to model signal processing applications as process networks. The implementation is not an issue though, yet problems emerge when addressing the integrity of the system under development. Designs that use indeterminate constructs are no longer sound by construction..

A merge operator captures nondeterminism in a fundamental way. A merge component has two input ports and a single output port. The internal indeterminate process selects events (tokens) from either input channel and copies them in order of selection onto the output channel. The merge operator is a common component in networks, either explicit or implicit ones.

As an example [due to Russell, 1990], consider an indeterminate process $\tilde{P}_1$ with a single input port and a single output port (Figure 5.5(a)). $\tilde{P}_1$ transforms the input sequence u into the output sequence y. $\tilde{P}_1$ internally has two deterministic modes of operation: 1) a read from the input port, followed by a write of a 0, followed by a write of a 1 to the output port, denoted as $R;0;1$. 2) a write of a 0 to the output port, followed by a read from the input port, followed by a write of a 0 to the output port, denoted as $0;R;0$.

In Figure 5.5(b) the indeterminate process $\tilde{P}_1$ has been divided into a determinate process P_1 (indicated by a sphere) and an indeterminate process $\tilde{M}$ (indicated by a box). The process P_1 resembles $\tilde{P}_1$ but is augmented with a control port. The mode of operation of P_1 is specified by the value read from the control port. If $c = 0$ then the mode of operation 1) is selected, and if $c = 1$ then the mode of operation 2) is selected. The control port of P_1 is driven from an indeterminate merge component $\tilde{M}$, that has two infinite input streams connected to its input ports. Stream u_0 contains events $\lambda_i = 0$ and stream u_1 contains events $\lambda_j = 1$. The merge component together with these infinite input streams makes up an *oracle*, which captures the indeterminate part of the process $\tilde{P}_1$.

The input-output relation of the network $\tilde{P}_1$ depends on the indeterminate selection mechanism and the *context* of the network. Given the possible choices for the mode of



(a) Non-deterministic process.

(b) Oraclised process.

Figure 5.5: Indeterminate processes.

Table 5.1: Input-output relations, given the control (c) and the (in)availability of input events (u). Note, S closed in Figure 5.6.

$\neg u$ denotes that u has **no** events ($\langle\perp\rangle$), whereas u denotes that u has events.

(a) $\tilde{P}_1$.

c	$\neg u$	u
0	$\perp$	0;1
1	0	0;0

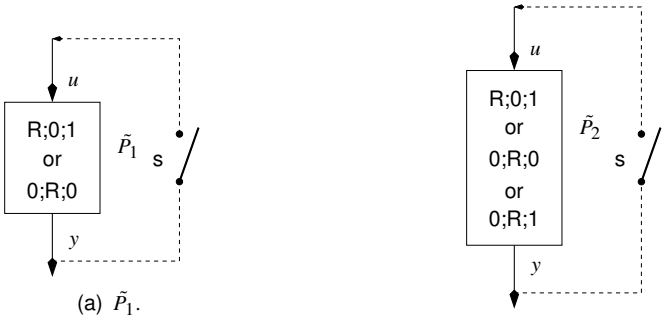
(b) $\tilde{P}_2$.

c	$\neg u$	u
0	$\perp$	0;1
1	0	0;0
2	0	0;1

operation and given the availability of events at the input port, the output events follow from evaluation; see Table 5.1(a). The input-output relations depend on the value of c and on the (in)availability of events on u (context).

One can construct a network $\tilde{P}_2$ with input-output relations that are equivalent to those of $\tilde{P}_1$. The input-output relations, however, do not solely determine the modes of operation. Let the possible modes of operation of $\tilde{P}_2$ be equal to those of $\tilde{P}_1$ with an additional mode of operation, namely: 3) a write of a 0 to the output port, followed by a read from the input port, followed by write of a 1 to the output port, denoted as $0; R; 1$. Obviously $\tilde{P}_1$ and $\tilde{P}_2$ have equivalent input-output relations; see Tables 5.1(a) and 5.1(b) respectively. Both processes generate either $\{\langle\perp\rangle, \langle 0\rangle\}$ or $\{\langle 0; 1\rangle, \langle 0; 0\rangle\}$, depending on the (in)availability of events on u.

With respect to the networks of Figures 5.6(a) and 5.6(b) we evaluated the input-output relations of $\tilde{P}_1$ and $\tilde{P}_2$ with the switch S opened. However, when operated in a different context (S closed); the two networks behave differently, and hence the input-output relation of indeterminate networks is not compositional. In [Russell, 1989; Panangaden, 1995] this phenomenon is exemplified by introducing a feedback link, thus closing the switch S in the networks of Figures 5.6(a) and 5.6(b). The input-output relations of these networks are given in Tables 5.2(a) and 5.2(b) respectively, and obviously are no longer equivalent. The above example illustrates the previously



(a) $\tilde{P}_1$. (b) $\tilde{P}_2$.

Figure 5.6: Process with context switch.

Table 5.2: Input-output relation with feedback (S closed in Figure 5.6).

$\perp$ denotes the empty sequence ($\langle\perp\rangle$).

(a) $\tilde{P}_1$.

c	
0	$\perp$
1	0;0

(b) $\tilde{P}_2$.

c	
0	$\perp$
1	0;0
2	0;1

mentioned Brock and Ackermann anomaly.

In Figure 5.5(b) we introduced an oracle that captures the indeterminate part of the network $\tilde{P}_1$. Nothing has been specified so far about the behaviour of the merge operation. It turns out that there exists a hierarchy of merge primitives ([Panangaden and Stark, 1988]) with provable inequivalent expressive power. The list of common merge primitives ranked according to their expressibility from strong to weak includes: *fair* merge, *angelic* merge, *infinity-fair* merge, and *unfair* merge.

Fairness is closely connected to nondeterministic or parallel program execution [Apt and Plotkin, 1986]. A classical example in this respect [due to Dijkstra, 1976, p.76] is the following parallel program; parallel (non-interleaved) execution of two statements S_1 and S_2 is denoted as $S_1 \parallel S_2$.

$$b := \textbf{true}; \quad x := 0;$$
$$\textbf{do } b \rightarrow x := x + 1; \parallel b \rightarrow b := \textbf{false}; \textbf{ od}$$

The program is guaranteed to terminate only under the assumption of true fairness. The actual value of the variable x at termination of the program depends on *when* the scheduler decides to invoke the right-hand-side statement ($b := \textbf{false}$). In case of an unfair scheduler, termination of the program is not guaranteed.

The merge primitive has two input ports and a single output port. Events (or tokens) are read from either input port and written unaltered onto the output port. The relative

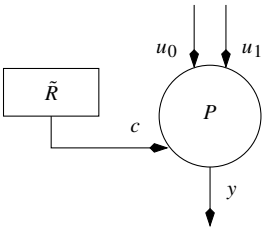


Figure 5.7: Infinity fair merge oraclised.

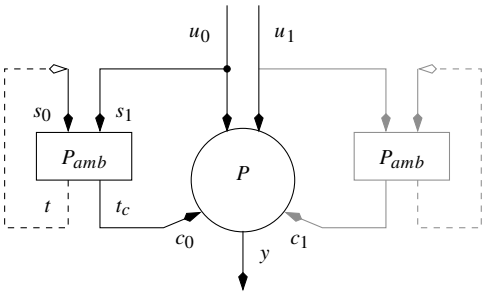


Figure 5.8: Angelic merge oraclised.

order of events of the input stream is preserved in the output stream. Let u_0, u_1 be the input sequences and y the output sequence. Let y_0 and y_1 be two streams constructed from the output stream y as follows: events of y that originate from u_0 go to y_0, events of y that originate from u_1 go to y_1. The order of events of u is preserved on y_0 and y_1.

The *fair* merge primitive guarantees that under all circumstances $y_0 = u_0$ and $y_1 = u_1$. The *angelic* merge transmits all events of stream u_0 if the stream u_1 is *finite* and it transmits all events of stream u_1 if the stream u_0 is finite. The angelic merge thus behaves like a fair merge if both u_0 and u_1 are finite. The *infinity-fair* merge is the dual of the angelic merge. The infinity-fair merge transmits all events of u_0 provided that u_1 is *infinite* and it transmits all events of u_1 provided that u_0 is infinite. Thus the infinity-fair merge behaves like the fair merge if both u_0 and u_1 are infinite. The *unfair* merge (the ordinary "or" operation, also known as bounded nondeterminism) finally, does not give any guarantees at all.

The mere name of the infinity-fair merge is due to the composition [Panangaden and Shanbhogue, 1992] of a deterministic component and an indeterminate random number generator. The random number generator acts as the oracle and generates a sequence of positive integers. The determinate process has the usual two input ports for streams u_0 and u_1 and an output port y. In addition the component is equipped with a control port c to which the generated random number stream is connected. The process starts with reading an integer from the control port. The control value specifies the quantity of tokens to be read alternately from u_0 and u_1. See Figure 5.7 for a diagram.

In Section 5.1.1 we found that the monotonic property of a process is a sufficient condition for the Kahn model of computation to be compositional. An oracle effectively captures the indeterminate part of an indeterminate primitive. Therefore, when context dependent relations can be made explicit, the compositional property is retained. In fact, an oracle can only be devised for processes that have a (weak) form of monotonic behaviour. Panangaden presents two weak forms of monotonicity: Hoare monotonicity and Smyth monotonicity [Panangaden, 1995]. Hoare monotonicity is a weaker type of monotonicity than Smyth monotonicity, which in turn is a weaker form of monotonicity than of that the definition we gave in Section 5.1.1. In [Panangaden, 1995], the fair merge primitive is identified as being non-monotonic, the angelic merge

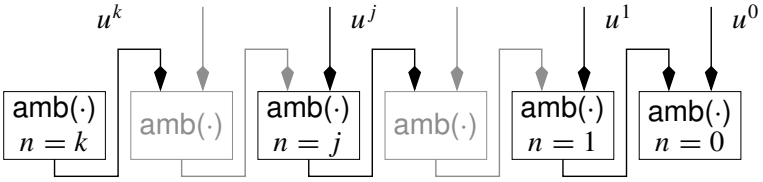


Figure 5.9: Countable non-determinacy using McCarthy's amb(·).

is identified as having Hoare monotonicity and infinity-fair merge has Hoare monotonicity as well as Smyth monotonicity.

The implementation of a non-monotonic function requires preemption of the computations, for the output stream is affected by non-causal events on the input stream. Preemption is commonly implemented by constructs like timeout, interrupt, or polling. A preemption-free implementation is possible, since angelic merge and infinity-fair merge behave in a sense monotonically. Fair merge, being non-monotonic, cannot be oraclised [Panangaden and Shanbhogue, 1988].

The oraclisation of the angelic merge primitive is not as straightforward as the infinity-fair merge primitive. In [Panangaden and Shanbhogue, 1988] a sketch of the composition for angelic merge is given. The indeterminate part (oracle) implements McCarthy's ambiguity operator amb(·) in a recursive network. "amb(x, y) has possible values x and y when both are defined: otherwise whichever is defined" [McCarthy, 1963]. A process that incorporates the amb(·) operator is a two-input, single-output node; the output sequence equals the non-empty input sequence; if both input sequences are non-empty then amb(x, y) returns y.

Recursive application of the amb(·) operator may be used to count the events of an input sequence. Consider a recursively defined function f:

$$f(n) = \mathsf{amb}(n, f(n+1)) .$$

As an example let the function less(K) return an empty event and have less(n) recursively defined as above. Then for any $n < K$, less(n) returns $K - 1$.

In order to count the number of events in a stream, we can apply the amb(·) operation as follows. Suppose the k-th event λ_k of stream u is available but λ_{k+1} is nonexistent (empty). Obviously if λ_k exists so does λ_j for $0 \leq j \leq k$. Thus $f(k)$ evaluates to k and subsequently $f(0)$ evaluates to k. Hence the number of events in u is counted as k, and therefore this type of nondeterminism is also known as countable nondeterminacy.

The fully expanded network (Figure 5.9) has k amb(·) operations: as many as the number of events in u. An projection of the one-dimensional array of k amb(·) operators [like in Kung, 1988] results in a process P_{amb} (Figure 5.8). The node P_{amb} implements the amb(·) operator and has two input ports, s_0 and s_1, and two output ports, t and t_c. P_{amb} maintains a state q. Initially, we have $q = 0$. The process implements an infinite loop. The body of the loop starts off by putting an event q on the port t. Next, P_{amb} evaluates its input ports. The process proceeds with one out of

three possible scenarios: 1) s_0 has an event but s_1 has no event; copy the event from s_0 to t_c and reset the state ($q = 0$). 2) Both s_0 and s_1 have an event; read and discard the events from s_0 and s_1 and increment the state ($q = q + 1$). 3) s_0 has no event, but s_1 has an event; read and discard the event from s_1 and set the state $q = 1$. Hereafter the body of the loop is re-invoked, sending an event q on t. The output t is fed back to the input s_0 of process P_{amb}. The input port s_1 is connected to the input stream u_0. The generated event that eventually is output on port t_c is a control signal to a deterministic process P. This control event is guaranteed to specify a count that never exceeds the remaining number of tokens of the stream u_0. A deterministic process can thus be devised that never blocks on stream u_0. The deterministic primitive requires equivalent control information for u_1 to implement angelic merge. Although amb$(\cdot)$ does embody a polling mechanism, it is impossible to implement the fair merge primitive with this construct [Panangaden and Shanbhogue, 1988].

We conclude with some remarks on the hierarchy of expressiveness of the merge primitives. There exists a composition of fair merge primitives that implement the angelic merge primitive. Further, an infinity-fair merge can be composed using angelic merge primitives. The reverse, however, is not possible. Above we exemplified the Brock and Ackermann anomaly. In their original paper [Brock and Ackermann, 1981], Brock and Ackermann prove the non-compositional property of input-output relations. The proof relies explicitly on the fairness of the applied (fair) merge primitive. Russell proved the non-compositional property for unbounded choice (unfair merge) [Russell, 1989] and thus the non-compositional property for any merge primitive. This result is supported by the observations in [Panangaden, 1995] that neither of the merge primitives is monotone in the definition of Section 5.1.1 and hence neither of the merge primitives is compositional.

5.3 CAPN semantics

Evidently, it has been recognised for long that the inclusion of some form of nondeterminism in any model of computation is beneficial for practical purposes. It greatly extends the scope of applications that can be modelled intuitively. On the other hand, introducing nondeterminism breaks the analytical strength of a model of computation. Keeping a shear balance is important.

In his seminal paper [Kahn, 1974], Kahn considered a possible indeterminate extension of the KPN semantics. He proposed a primitive process, named warn$(\cdot)$. The warn$(\cdot)$ process has two inputs and a single output. A *true* event is sent on the output whenever an event is received at either of the input ports. In contrast to McCarthy's amb$(\cdot)$ operation, Kahn's warn$(\cdot)$ operation does not pass incoming events but generates new (true valued) events.

In this section, we introduce two types of primitives. A set of convenience constructs for simulation purposes and a *register* link, which brings in nondeterminism.

5.3.1 Synchronising constructs

Kahn semantics consider process networks from a viewpoint that concentrates on the history of events. Another approach is consider the same process network from a view-

point that concentrates on the state changes of the respective processes. One example is the observer semantics introduced in the 1930s by Muller, named cumulative states [Miller, 1965, Chap. 10]. Although a change of viewpoint changes the focus of concerns, it does not influence the properties of a network.

A commonly applied primitive for condition synchronisation (Definition 5.1–i) in asynchronous communication systems is the Muller–C [Miller, 1965; Wuu, 1993] element. This basic element is a two-input port device with a single output port. The Muller–C element can be viewed as the logical AND of two events. In integrated circuits (IC) an event is usually a $0-1$ or $1-0$ transition. The output thus equals the inputs after both inputs reach the same value, otherwise the output keeps (latches) the previous output.

Kahn semantics include condition synchronisation, hence the implementation of a Muller–C equivalent process is straightforward. A Muller–C process in Kahn semantics is a process with two input ports and one output port. The process outputs a true valued event as soon as two events have been read, one from each input port. A simple continuous process with one mode of operation suffices. First, read from all input ports in a predefined order while discarding the events. Then, output a true valued event on the output port.

A Muller–C element with more than two input ports can easily be composed using a tree structure of basic Muller–C elements. It requires $(n-1)$ Muller–C elements to devise an n-input Muller–C element. A tree structure, albeit functionally sound, is not the most efficient implementation. In Kahn semantics a Muller–C process can simply be generated from a template process, in which the number of input ports is a parameter. For IC designs, dedicated structures are applied [see e.g. Wuu, 1993].

The dual of the Muller–C element is Kahn's $\mathsf{warn}(\cdot)$ process. We refer to a parameterised version of the $\mathsf{warn}(\cdot)$ process as an N-to-one semaphore ($\mathrm{Sem}_{N\to 1}$). The basic $\mathrm{Sem}_{2\to 1}$ ($\mathsf{warn}(\cdot)$) process has two input ports and a single output port. The process outputs a true valued event whenever an event is read from either of the input ports. The output event thus specifies the availability of an event but does not specify at which input port the event was received. An extension to a process with N-input ports is straightforward.

The practical value of a $\mathrm{Sem}_{N\to 1}$ is limited to an executable model of the Kahn MoC. When applied in the composition of Figure 5.10, the control value c merely offers a *trigger* to a process of the reception of an event. The control event triggers a ready-to-run state of the process, in a similar way as the mechanism of the demand-driven schedule sketched in Section 5.1.1.

Like the Muller–C element, the $\mathrm{Sem}_{N\to 1}$ implements condition synchronisation. Compositions that use either of these elements are still determinate.

5.3.2 Register

A register is a unidirectional channel (REG link) between an input port and an output port. Unlike an unbounded FIFO link, a register link has bounded capacity: it is a one-place buffer. Operations performed on a REG link are never blocking. A write to a REG link will overwrite the current value (state) of the channel. A read from a REG reports

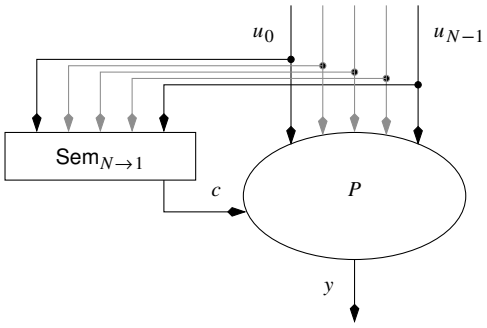
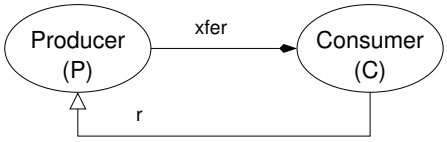


Figure 5.10: N-to-one semaphore process.



xfer: FIFO link (solid arrow) and r: REG link (open arrow).

Figure 5.11: Producer-consumer network with REG link feedback.

the current state. These behaviours are sometimes referred to as overwrite-on-full and last-on-empty-read, respectively.

As an example, consider a typical producer-consumer example with a REG feedback link (Figure 5.11). In this diagram and following ones, FIFO links are indicated with a solid arrows and REG links are indicated with open arrows. Each invocation (firing) of the producer P will trigger a get$(\cdot)$ from the control channel r followed by a put$(\cdot)$ on the data channel xfer. The actual value of the event on the FIFO link xfer depends on the previously read control event. Similarly, each invocation of the consumer C will trigger a get$(\cdot)$ from the data channel and a corresponding put$(\cdot)$ on the control channel. Depending on the relative schedule of the producer and the consumer, control events sent by the consumer may be transferred, replicated, or discarded.

To show the schedule-dependent behaviour of a register channel, consider the following. Let λ_0 be the register value of the REG link after reset and $\langle \lambda_1, \lambda_2, \cdots \rangle$ be the sequence of coordination events emitted by the consumer on successive invocations. If a schedule alternately invokes the producer and consumer, the producer observes the same sequence of coordination events: $\langle \lambda_0, \lambda_1, \lambda_2, \cdots \rangle$. An alternative schedule that alternately invokes the producer k times followed by k successive invocations of the consumer lets the producer observe $\langle (\lambda_0)^k, (\lambda_k)^k, (\lambda_{2k})^k, \cdots \rangle$. That is, k replications of λ_0 followed by k replications of λ_k, while intermediate coordination events are discarded. A fully parameterised schedule demonstrates the context dependency of the network even further. Let $S = \langle \langle \rho_0^p, \rho_0^c \rangle, \langle \rho_1^p, \rho_1^c \rangle, \cdots \rangle$ describe the sequence of in-

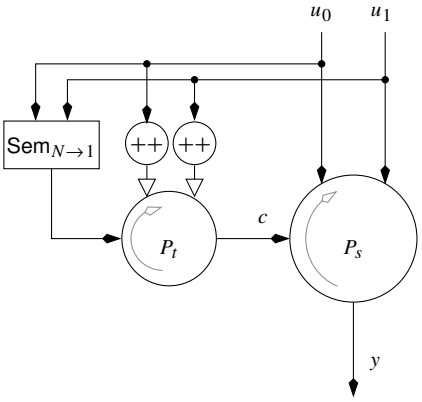


Figure 5.12: Select compound.

vocations of each process. First the producer is invoked ρ_0^p times, then the consumer ρ_0^c times, followed by the producer ρ_1^p times, etc. As a result the producer observes the following control sequence:

$$\langle (\lambda_0)^{\rho_0^p}, (\lambda_{\rho_0^c})^{\rho_1^p}, (\lambda_{\rho_0^c+\rho_1^c})^{\rho_2^p}, \cdots, (\lambda_{\sum_{s=0}^{i-1}\rho_s^c})^{\rho_i^p} \rangle, \tag{5.3}$$

of which the previous examples are just special cases, i.e., $\rho_i^p = \rho_i^c = k$ for all i.

5.4 Analysis and application of CAPN

In this section, we analyse the potentials of the $\text{Sem}_{N\to 1}$ and REG primitives introduced in the previous section. Given these primitives, one can construct a useful primitive in practical situations: the select primitive. The select compound was introduced in Section 5.2 as a construct to model user interference with the network. The behaviour, i.e., the fairness of the constructed select compound will depend on the applied schedule of the network. In this section we will show that the register has the necessary monotonic properties to allow for oraclisation.

5.4.1 Select compound

Given the $\text{Sem}_{N\to 1}$ and REG primitives a *select* compound is constructed. The select compound is functionally equivalent with the select primitive proposed in [de Kock et al., 2000]. Our select compound is constructed out of determinate (Kahn) processes, FIFO links, and indeterminate REG links. In order to analyse the compositional property of the select compound it is sufficient to study its indeterminate primitives. In this case all indeterminism is isolated in the applied REG links.

The select compound implements a merge process. Given two input streams u_0 and u_1, the merge process transfers events from u_0 and u_1 to an output stream y. The

relative order of events from each input stream is preserved in the output stream, yet the interleaving of events originating from u_0 and u_1 is left unspecified.

The construction of the select compound is given in Figure 5.12. The network is a Kahn process network but one with two channels: the channels from the counting processes (++) to the process P_t are implemented as REG links. In the diagram, each incoming stream u_0 and u_1 is *forked* to a counting process (++), an $\text{Sem}_{N\to 1}$ process, and the main selection process (P_s). A counting process node implements an infinite loop process. The body of the loop starts with invoking a get$(\cdot)$ from the input port, then increments a counter value, and concludes with the invocation of a put$(\cdot)$ of the actual counter value on the output port of the process node. The $\text{Sem}_{N\to 1}$ process node outputs a true valued event for every event received at either of its input ports. The process P_t synchronises on the output of the $\text{Sem}_{N\to 1}$ process: P_t awaits (get$(\cdot)$) an event from the $\text{Sem}_{N\to 1}$. The arrival of an event indicates the availability of an event on either u_0 or u_1. Subsequently P_t will read (get$(\cdot)$) each of the REG links and compare their value with the stored, most recently read, value. If a process would read all subsequent events from a counting process, then the process would observe a sequence of increasing integers. Thus by maintaining state, P_t can observe discrepancies between the current and previous state of the register channel. Any discrepancy indicates the availability of an input event. In case of a detected input event P_t emits a control event on c that specifies the input port, u_0 or u_1 with an pending event. The selection process P_s reads (get$(\cdot)$) an event from its control port c and subsequently issues a get$(\cdot)$ on the specified port, u_0 or u_1. The read event is copied on the output stream y.

The select compound of Figure 5.12 is naturally divided into a determinate part (P_s) and an indeterminate oracle. The indeterminate character of the oracle is concentrated in the two REG links from the two counting processes to P_t.

5.4.2 Register oracle

The behaviour of the REG link is indeterminate, yet it is a form of controlled nondeterminacy. Recall the producer-consumer network of Figure 5.11. As shown in Equation (5.3), the effective control sequence r of this network is parameterised. It solely depends on the applied schedule of the network.

In this section, we devise an oracle that captures the indeterminate character of the REG link. To illustrate the oracle, we first examine a static configuration and then present a dynamic configuration. The dynamic configuration concentrates the schedule (context) dependent behaviour of the REG link in a control stream.

The input-output relations of a REG link are comparable to the input-output relations of sample-rate conversion (SRC) constructs. These SRCs have proven their usefulness in the field of multirate digital signal processing. A sample-rate converter pairs an *interpolator* (up sampler) and a *decimator* (down sampler). Figure 5.13 shows a sample-rate converter in which a decimator is followed by an interpolator.

In digital signal processing it is common to implement so-called zero-filling interpolation [Crochiere and Rabiner, 1981]. A zero-filling interpolator reads events from the incoming stream and outputs a stream which is an exact copy of the incoming stream, but, with L zeros inserted between successively read events. Here we take a slightly different approach. Instead of inserting zeros we insert L copies of the last

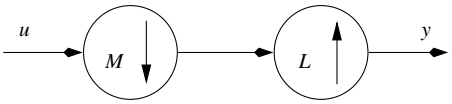


Figure 5.13: Decimator-Interpolator.

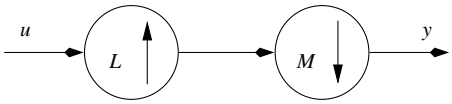


Figure 5.14: Interpolator-Decimator.

read event. Thus given an input sequence of $\langle \lambda_0, \lambda_1, \cdots \rangle$ the interpolator outputs

$$\text{Interpolator}(L): \quad \langle (\lambda_0)^L, (\lambda_1)^L, \cdots \rangle. \tag{5.4}$$

A decimator outputs a down-sampled version of the input stream. That is, out of M read events all but one event are discarded; the last one is written to the output stream. Again, given an input sequence of $\langle \lambda_0, \lambda_1, \cdots \rangle$ the decimator outputs

$$\text{Decimator}(M): \quad \langle \lambda_{M-1}, \lambda_{2\,M-1}, \cdots, \lambda_{k\,M-1}, \cdots \rangle. \tag{5.5}$$

We arbitrarily chose to copy the last element of a block of M input events to the output port. Any other element of the block of M events could have been chosen as output, however. A proper initialisation of the channel is sufficient to accomplish a shift of the selected element. To illustrate this, assume that we initialise the input stream with $M-1$ void events. The resulting output stream would then be

$$\langle \lambda_0, \lambda_M, \cdots, \lambda_{k\,M}, \cdots \rangle,$$

which is merely a shift of the sequence of Equation (5.5).

A sample-rate converter (SRC) combines a decimator and an interpolator. In Figure 5.13 a stream u is processed by a decimator followed by an interpolator. This is a static, yet parameterised (M, L), SRC. Given an input sequence $u = \langle \lambda_0, \lambda_1, \cdots \rangle$, the output sequence $y = \mathsf{DecInt}(M, L; u)$ follows from subsequent application of Equation (5.5) and Equation (5.4).

$$\mathsf{DecInt}(M, L; u): \quad y = \langle (\lambda_{M-1})^L, (\lambda_{2\,M-1})^L, \cdots, (\lambda_{k\,M-1})^L, \cdots \rangle \tag{5.6}$$

A configuration of a sample-rate converter with a decimator and an interpolator in reversed order yields a slightly more complicated expression for the output sequence. In Figure 5.14 a stream u is processed by an interpolator followed by a decimator. Given an input sequence $u = \langle \lambda_0, \lambda_1, \cdots \rangle$, the output sequence $y = \mathsf{IntDec}(L, M; u)$, now, has replicated consecutive events as follows:

$$\mathsf{IntDec}(L, M; u): \quad y = \{ (\lambda_0)^{\varphi[0]}, (\lambda_1)^{\varphi[1]}, \cdots \}. \tag{5.7}$$

The replication factor $\varphi[j]$ is periodic. It is recursively defined as:

$$\begin{array}{lcl} \varphi[j] & = & \left\lfloor \frac{\theta[j]}{M} \right\rfloor \\ \theta[0] & = & L \\ \theta[j+1] & = & L + \theta[j] \bmod M \end{array} \tag{5.8}$$

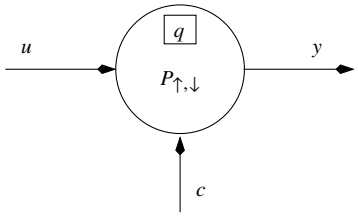


Figure 5.15: Sample-rate converter (SRC).

for $j = 0, 1, \cdots$. The replication factor $\varphi[j]$ (and thus $\theta[j]$) has a period N: $\varphi[n + N] = \varphi[n]$. Not surprisingly, N depends on the greatest common divider of L and M: N = $M/\text{gcd}(L, M)$.

A generic sample-rate converter (SRC) dynamically combines the networks of Figures 5.13 and 5.14. Figure 5.15 shows a sample-rate converter with explicit (parameterised) control. The control stream dictates the topology of the SRC (a decimator followed by an interpolator or vice versa), it dictates the respective sample rates (L and M), and it dictates changes from one configuration (topology and sample-rate) to another. The sample-rate converter operates as follows. The control stream c has a sequence of events $\langle \alpha_0, \alpha_1, \alpha_2, \cdots \rangle$, $\alpha_i \in \{-1, +1\}$. A negative control value ($\alpha_j = -1$) corresponds to the invocation of a decimator ($M = 1$); a positive control value corresponds to the invocation of an interpolator ($L = 1$). The process $P_{\uparrow,\downarrow}$ starts with reading a control value α_j from the control port. A negative control value triggers the read of an event from port u. The event is stored in the status register ($q \leftarrow \mathsf{get}(\cdot)$). A positive control value triggers the write of a copy of the stored event $y \leftarrow \mathsf{put}(q)$. Formally, let $q = \lambda_{-1}$, $u = \langle \lambda_0, \lambda_1, \lambda_2, \cdots \rangle$, and $y = \langle \perp \rangle$ be the current state of the SRC network (Figure 5.15). The state of the network after a read of the control value α_0 is as follows:

$$\alpha_0 = -1 \begin{cases} u = \langle \lambda_1, \lambda_2, \cdots \rangle \\ q = \lambda_{0|} \\ y = \{\perp\} \end{cases} \qquad \alpha_0 = +1 \begin{cases} u = \langle \lambda_0, \lambda_1, \cdots \rangle \\ q = \lambda_{-1} \\ y = \{\lambda_{-1}\} \end{cases}$$

In Section 5.4.3 we further analyse the properties of the control sequence c, with in particular the relation between c and the context of the network. The behaviour of the Interpolator (Equation (5.4)) and the Decimator (Equation (5.5)) can be described by their respective control streams c_I and c_D. Let the state of the SRC be defined as before: $q = \lambda_{-1}$, $u = \langle \lambda_0, \lambda_1, \lambda_2, \cdots \rangle$ and $y = \langle \perp \rangle$, the control streams c_I and c_D are as follows:

$$\text{Interpolator}(L): \quad c_I = \langle -1, (1)^L, -1, (1)^L, -1, \cdots, \rangle \tag{5.9}$$

$$\text{Decimator}(M): \quad c_D = \langle (-1)^M, 1, (-1)^M, 1, \cdots \rangle \tag{5.10}$$

As a further example, let the state of the SRC be as before. Consider a control sequence $c = \langle -1, (1)^k, (-1)^k, (1)^k, (-1)^k, \cdots \rangle$. Then the output stream yields $y =$

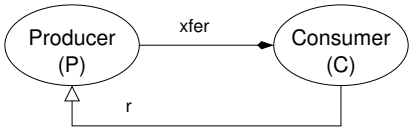


Figure 5.16: CAPN Producer-consumer network.

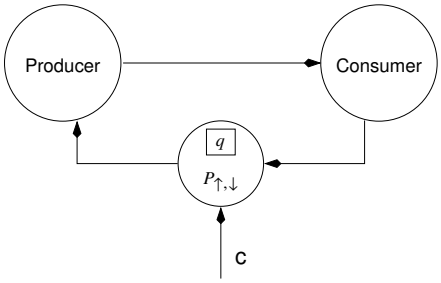


Figure 5.17: KPN Producer-consumer network with register model.

$\langle(\lambda_0)^k, (\lambda_k)^k, (\lambda_{2k})^k, \cdots\rangle$, which corresponds to the result of Section 5.4.2. More generally, consider a control stream $c = \langle -1, (1)^{\rho_0^p}, (-1)^{\rho_0^c}, (1)^{\rho_1^p}, (-1)^{\rho_1^c}, \cdots\rangle$ where $\rho_i^p, (\rho_i^c \in \mathbb{N})$. With the current state of the network as before, y is given as:

$$y = \langle(\lambda_0)^{\rho_0^p}, (\lambda_{\rho_0^c})^{\rho_1^p}, \cdots, (\lambda_{\rho_0^c+\rho_1^c})^{\rho_2^p}, \cdots, (\lambda_{\sum_{s=0}^{i-1}\rho_s^c})^{\rho_i^p}\rangle \tag{5.11}$$

which corresponds to Equation (5.3).

The SRC effectively models a REG link, provided that an appropriate control stream can be generated. The network of Figure 5.16 is a copy of the producer-consumer network of Figure 5.11 on Page 97. In the network Figure 5.17 the REG link of Figure 5.16 has been replaced with a SRC, which leaves a KPN model of a CAPN. The context-dependent behaviour of the REG link is now confined in the control stream c. The remaining issue, thus, is how to generate a control stream with appropriate properties for the network of Figure 5.17. In the remainder of this section we consider two approaches. The first approach uses an angelic merge construct to generate c dynamically, the second approach considers a predetermined cyclo-static control stream. The first approach necessarily is non-deterministic whereas the second approach is deterministic and therefore also compositional. In the next section (Section 5.4.3) we pursue an analytical approach.

Non-deterministic control stream

A possible emulation of the REG link uses an angelic merge to generate a control stream for the SRC of Figure 5.17, see Figure 5.19 for the corresponding network. The resulting behaviour of this emulated REG link depends entirely on the applied schedule.

The REG link emulation relates to Parks's method of establishing feasible schedules of Kahn process networks using bounded memory implementations rather than unbounded memory [Parks, 1995]. Parks's method is based on the emulation of a bounded FIFO through a *pair* of unbounded FIFOs with a corresponding access protocol. The diagram of Figure 5.18 shows the principle applied in a common producer-consumer setting. Normally, there is a single unbounded FIFO in the network, from

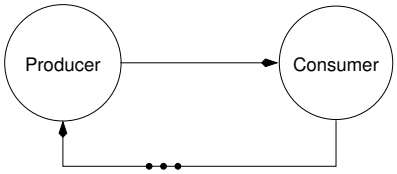


Figure 5.18: Bounded FIFO emulation.

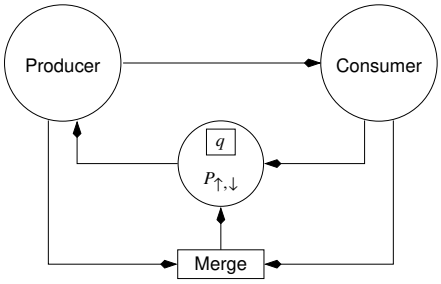


Figure 5.19: REG link emulation.

producer to consumer. In order to limit the utilisation of the buffer capacity of this link, a second (unbounded) control FIFO is added from the consumer back to the producer. This second channel is initialised with k events, and in addition access protocol of the producer and consumer are modified. Before the producer emits (put$(\cdot)$) an event to the consumer, the producer has to read an event from the control FIFO. After the consumer receives (get$(\cdot)$) an event from the producer, the consumer puts an event on the control FIFO. This way, the number of events that concurrently resides in the FIFO links never exceeds k, since either the producer or the consumer will encounter a blocking read due to the unavailability of events.

The emulation of a REG link applies a comparable method: replacing the REG link with a SRC, an angelic-merge construct, and a modification of the access protocol. Figure 5.19 gives the diagram of the REG link emulation. The access protocol of the REG link as implemented by the producer and consumer is changed as follows. Before the producer reads (get$(\cdot)$) the current register value, it generates an event $\alpha = 1$ to the merge construct. Similarly, before the consumer updates (put$(\cdot)$) the current register value, it generates an event $\alpha = -1$ to the merge construct. The merge construct passes the incoming events to the SRC. The SRC behaves as specified before. As a result, the network shows the same schedule (context) dependent behaviour we saw in Section 5.3.2. Angelic merge is the weakest possible form of the merge primitive that guarantees a deadlock-free implementation. Suppose the consumer applies a finite number of updates, then all consecutive producer requests should be acknowledged. Vice versa, if the producer poses a finite number of get$(\cdot)$ requests, then the consumer should be able to succeed all its consecutive put$(\cdot)$ operations. Angelic merge guarantees precisely this (see Section 5.2).

The effective oraclisation of the REG link implies a form of monotonicity. Since the SRC is a monotonic process, the REG link emulation shares the (weak) form of monotonicity with the angelic-merge primitive (Hoare monotonicity).

Cyclo-static control stream

In case of a deterministic context of the network, the control stream of the SRC of Figure 5.17 is also deterministic. We propose to generate such a stream through the use of

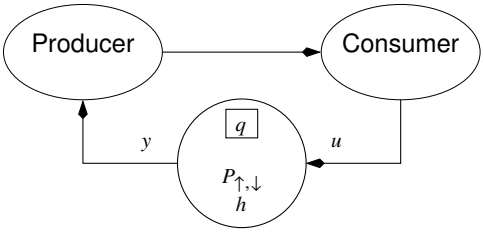


Figure 5.20: Cyclo-static REG link emulation ($y = \mathsf{SRC}(h; u)$).

a parameterised sample rate converter that builds upon the parameterised Interpolator and Decimator processes. The Interpolator, Decimator, and their compositions have a regular cyclic behaviour. We capture this behaviour by using a converter process with an associated characteristic set of coefficients (h) (Figure 5.20). This set h describes a sample rate conversion pattern, which is comparable to the kernel of a convolution process. The canonical form of h uniquely characterises the converter process, moreover h is compositional.

We denote the converter process as $y = \mathsf{SRC}(h; u)$. The coefficients of h_i of h define the replication factor ($h_i \in \mathbb{N}$) of an input data token in the output data stream y; h is applied cyclically to the input stream u. Let $u = \langle \lambda_0, \lambda_1, \lambda_2, \cdots \rangle$ be the input data stream and $h = \{h_0, h_1, \cdots h_{N-1}\}$ be the characteristic (ordered) set of coefficients with period N, then the output sequence y of $\mathsf{SRC}(h; u)$ is given as:

$$y = \langle (\lambda_0)^{h_0}, (\lambda_1)^{h_1}, \cdots, (\lambda_{N-1})^{h_{N-1}}, \cdots, (\lambda_i)^{h_{i \bmod N}}, (\lambda_{i+1})^{h_{(i+1) \bmod N}}, \cdots \rangle. \tag{5.12}$$

As an example, the output sequences of the Interpolator and Decimator, Equations (5.4) and (5.5) respectively, are generated by applying their respective coefficients h_I and h_D to an input sequence u; h_I and h_D are parametrically given as:

$$\text{Interpolator}(L): \quad h_I = \{L\} \tag{5.13}$$

$$\text{Decimator}(M): \quad h_D = \{(0)^{M-1}, 1\} \tag{5.14}$$

The length (dimension) of the characteristic set of coefficients, denoted $|h|_0$, determines the size of the block of input tokens the sample rate converter takes per cycle. The usual ℓ_1 vector norm of the characteristic set of coefficients, denoted $|h|_1$, determines the number of tokens in the output for every input block; $|h|_p = \sum_{i=0}^{N-1} (h_i)^p$ for $p \in \{0, 1\}$. With respect to Equations (5.13) and (5.14) we have: $|h_I|_0 = 1$, $|h_I|_1 = L$, $|h_D|_0 = M$ and, $|h_D|_1 = 1$.

There is a direct relation between the characteristic set of coefficients h of Fig-

ure 5.20 and the cyclically extended control stream c of Figure 5.17, namely:

$$\text{if } h_i = 0, \text{ then } \alpha_i = \langle -1 \rangle$$
$$\text{if } h_i > 0 \text{ then } \alpha_i = \langle -1, (1)^{h_i} \rangle$$

The cyclo-static control stream that drives the sample rate converter is a proper reasoning framework for further analysis. In the next subsection we derive a procedure for composing two cyclo-static control streams . The result yields again a cyclo-static control stream, which is uniquely characterised by a characteristic set of coefficients. In Section 5.4.3 we will apply our reasoning framework.

Compositionality

Cyclo-static control streams are compositional. The composition of two finite characteristic sets of coefficients f and g yields a finite set $h = g \circ f$. In order to form their composition f and g must be *balanced*: f and g are balanced iff $|f|_1 = |g|_0$, i.e., the number of outputs per cycle from $\mathsf{SRC}(f; u)$ matches the number of inputs per cycle of $\mathsf{SRC}(g; u)$ for any u. The balance condition can always be met because the cyclic extension of any finite h by a factor k yields equivalent behaviour of the converter process, $\mathsf{SRC}(h; u) = \mathsf{SRC}((h)^k; u)$, and f and g are finite.

Given two balanced finite characteristic sets of coefficients f and g their composition is found through the following procedure. Partition the set g according to the value of the subsequent entries of f. Let f_i be the i-th entry in f and g'_i be the corresponding i-th sub partition of g then due to the partitioning scheme we have $|g'_i|_0 = f_i$. In case $f_i = 0$, the corresponding g'_i equals a set with empty elements only, $\{\perp\}$, which has expected properties: $|\{\perp\}|_0 = |\{\perp\}|_1 = 0$. Consequently we define h through its entries as: $h_i = |g'_i|_1$. The partitioning of g is guaranteed to fit since $|f|_1 = |g|_0$. Observe that the resulting characteristic set of coefficients inherits the properties of its constituents: $|f|_0 = |h|_0$ and $|g|_1 = |h|_1$. Moreover h is finite by construction.

As an example consider the composition $h = g \circ f$. Let $f = \{5\}$ and $g = \{0, 0, 1\}$. Thus h is the characteristic set of coefficients of an $\mathsf{IntDec}(L = 5, M = 3)$. We first balance f and g; let $r = \gcd(|f|_1, |g|_0)$, we extend f and g cyclically as follows

$$\{(f)^{\frac{|g|_0}{r}}\} = \{5, 5, 5\},$$
$$\{(g)^{\frac{|f|_1}{r}}\} = \{0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1\}.$$

Subsequently we partition the cyclically extended g according to extended f and form

$$g' = \{\{0, 0, 1, 0, 0\}, \{1, 0, 0, 1, 0\}, \{0, 1, 0, 0, 1\}\}.$$

Taking the ℓ_1-norm of the subsets yields

$$h = \{1, 2, 2\},$$

which is equivalent with the result of Equation (5.7). The period of h, $|h|_0$ is equal to the cyclic extension factor of f, $\frac{|g|_0}{r} = \frac{M}{\gcd(L,M)}$. Further the number of outputs

of h, $|h|_1$, is equal to the cyclic extension factor of g, $\frac{|f|_0}{r} = \frac{L}{\gcd(L,M)}$. Hence, the composition $h = g \circ f$ of an Interpolator f and a Decimator g only depends on the ratio L/M; the norms $|h|_0$ and $|h|_1$ do not change if we multiply L and M with a factor $k \in \mathbb{N}^+$.

It is straightforward to derive the characteristic set of coefficients for a generic DecInt$(M, L; u)$ composition. Let $f = \{(0)^{M-1}, 1\}$ and $g = \{L\}$ be the characteristic set of coefficients for an Decimator and Interpolator, respectively. In this case f and g are balanced and g' is readily formed

$$g' = \{(\{\perp\})^{M-1}, \{L\}\},$$

which yields

$$h = \{(0)^{M-1}, L\}.$$

5.4.3 Analysis

A Kahn process network has arbitrarily many feasible schedules. Since KPNs are determinate, every schedule yields equivalent (functional) input-output relations of the network. The difference between alternative schedules is of a nonfunctional nature. An significant difference, for instance, is the memory usage of alternative schedules. Let $\mathcal{S}$ be the set of feasible schedules. Practical implementations try and generate a schedule $s \in \mathcal{S}$ with convenient (nonfunctional) properties. For instance, in [Parks, 1995], schedules are generated that can execute with bounded memory. In [Lieverse et al., 2001], schedules are generated based on a discrete event simulation of an architecture to which the Kahn process network is a workload. These latter schedules are consequently context aware. They depend on the timing constraints of a model of architectures, i.e., the typical delay of an architecture component. Hence, these schedules execute in bounded time.

Least-fixed point schedules are an important class of schedules [Lee and Parks, 1995]. Let the state of a network be defined by the number of events that resides in every channel of the network. Starting from state $\boldsymbol{q}$, the network returns to $\boldsymbol{q}$, after application of a least-fixed point schedule. Moreover the state traversal during the application of the schedule is minimal. Unfortunately, finding a fixed point schedule is in general impossible. Successful attempts are known for certain classes of dataflow networks [Najjar et al., 1999]. Analytical solutions are known for the static dataflow (SDF) case and its cyclic extension (CSDF). The dynamic dataflow (DDF) case only gives satisfactory solutions for a limited number of classes [Buck, 1993]. Note that the REG link emulation scheme of Figure 5.17 is a typical DDF process node.

In the remainder of this section, we derive a KPN network with parameterised, and thus asynchronous, coordination. The overall behaviour of the network depends on the actual setting of the parameters. We show in an example that in the case of static dataflow it is possible to derive a closed expression under which proper behaviour of the network is guaranteed. The section is concluded with a second example, which supports the case for using CAPN as a useful MoC for context-aware applications.

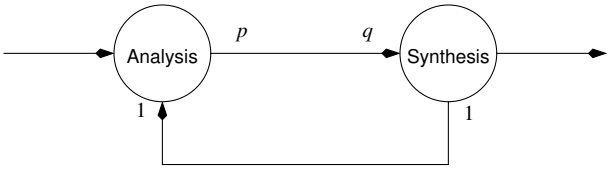


Figure 5.21: SDF synchronous cooperative network.

Asynchronous coordination

On several occasions we emphasised the importance of asynchronous coordination. Consider Figure 5.21, where a common synthesis-analysis network is shown with a cooperative synthesis process. The diagram presumes the so-called synchronous data-flow (SDF) MoC. A node in an SDF network specifies the number of tokens it reads or writes on every invocation (schedule) of the node. As a consequence the coordination of the synthesis and analysis processes is strictly synchronous (coherent). The analysis process requires a cooperation event for every generated workload event. Similarly, the synthesis process must send a cooperation event for every analysis event read. SDF networks are usually evaluated by means of an incidence or topology matrix Γ, [Lee and Messerschmitt, 1987]. The rows of an incidence matrix Γ correspond to a channel; the columns correspond to a node. The non-zero entry $\gamma_{i,j}$ specifies the production or consumption of tokens ($\gamma_{i,j} > 0$ respectively $\gamma_{i,j} < 0$) on channel i when node j is invoked (fired). The right-hand-side null space $\boldsymbol{\rho}$ of Γ ($\Gamma\boldsymbol{\rho} = \mathbf{0}$) equals the least-fixed-point cumulative schedule. That is, the entries ρ_j of $\boldsymbol{\rho}$ specify the cumulative number of invocations for each node j in a feasible schedule $s \in \mathcal{S}$ such that the network returns to its initial state. The right-hand side null space is a necessary condition for the consistency of the network, however insufficient for guaranteeing the existence of a valid static schedule [Bilsen et al., 1996].

The incidence matrix of the network of Figure 5.21 is given in Equation (5.15).

$$\begin{bmatrix} p & -q \\ -1 & 1 \end{bmatrix} \tag{5.15}$$

A non-trivial (one-dimensional) right-hand side null space exists only if Γ is singular, in fact it is known that rank(Γ) equals the number of links in the network minus one. The network is consistent if the determinant of Γ is zero, i.e., $p = q$. Thus we have

$$\begin{bmatrix} p & -p \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \mathbf{0}, \tag{5.16}$$

which specifies the cumulative schedule but a constant multiplication. This scheme implies synchronous coordination as p and q are tightly coupled.

In order to facilitate asynchronous coordination we statically decouple the analysis and synthesis processes with a sample rate converter scheme as depicted in Figure 5.22. The characteristic set of coefficients h defines the cyclo-static behaviour of $P_{\uparrow,\downarrow}$, turning the network into a CSDF network. The cyclo-static description of $P_{\uparrow,\downarrow}$ specifies at

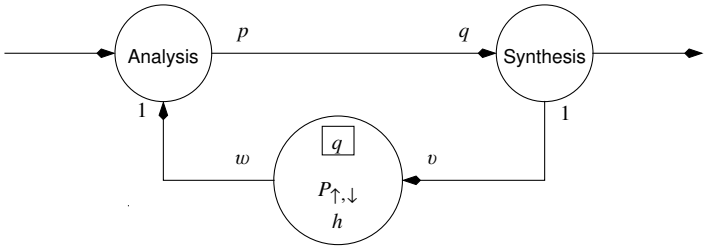


Figure 5.22: CSDF static asynchronous cooperative network.

its input port a sequence of $|h|_0 \times 1$ and sequence h at its output port; the period of $P_{\uparrow,\downarrow}$ equals $|h|_0$. The producer and consumer have a period equal to 1. The corresponding incidence matrix is constructed using the total number of samples per period [Bilsen et al., 1996], which corresponds to $v = |h|_0$ at the input port and $w = |h|_1$ at the output port. The incidence matrix is then as follows:

$$\begin{bmatrix} p & -q & \\ & 1 & -v \\ -1 & & w \end{bmatrix} \tag{5.17}$$

The right-hand-side null space of Γ is non-trivial if $p\,w = q\,v$. Let $r = \gcd(p, q)$ and $w = k\frac{q}{r}$, $v = k\frac{p}{r}$ then we have

$$\begin{bmatrix} p & -q & \\ & 1 & -k\frac{p}{r} \\ -1 & & k\frac{q}{r} \end{bmatrix} \begin{bmatrix} k\frac{q}{r} \\ k\frac{p}{r} \\ 1 \end{bmatrix} = \mathbf{0},$$

which effectively decouples the coordination by a (parameterised) factor of $\frac{w}{v}$.

Context dependency

In order to derive a closed expression for the context dependency of the cyclo-statically parameterised network of Figure 5.22, we require execution details of the processes in the nodes. Without this information only exhaustive simulation of the network, for all relevant values of h, can be used to analyse the correct behaviour of the network.

In this section we consider the network of Figure 5.22 in some more detail, adding enough execution details for circumventing lengthy simulations for all kind of characteristic sets of coefficients. We safely assume that the synthesis process supplies a control token to the analysis process to request a particular mode of operation of the analysis process. The synthesis process is obviously best served when the analysis process follows these requests seamlessly. However this is generally not the case, because of the differences in their relative rate at which the synthesis process supplies (q) and the analysis process acknowledges (p) mode change requests and the characteristics of the sample rate converter in the feedback loop. With this interpretation we focus

on the control. Consequently, the analysis-synthesis network equals an Interpolator-Decimator network with corresponding characteristic sets of coefficients: $g = \{p\}$ for the producer and $f = \{(0)^{q-1}, 1\}$ for the consumer. The entire network can be analysed through the composition $s = h \circ g \circ f$.

From Section 5.4.2 we already know that the composition $g \circ f$ only depends on the ratio $\frac{q}{p}$. Without loss of generalisation we presume p and q to be co-prime: $\gcd(p, q) = 1$. Since we consider a consistent network $\frac{q}{p} = \frac{w}{v}$ we also have a consistent closed network: $|s|_0 = |s|_1$. In the sequel we consider characteristic sets of coefficients for the sample rate converter where $|h|_0 = v = k\,p$ and $|h|_1 = w = k\,q$, for some $k \in \mathbb{N}^+$.

Suppose the feedback system implements a Decimator followed with an Interpolator: $h = \mathsf{DecInt}(k\,p, k\,q)$. In this case there is an closed form of $s = h \circ g \circ f$:

$$\begin{aligned} s &= \{(0)^{k\,p-1}, 1\} \circ \{k\,q\} \circ \{p\} \circ \{(0)^{q-1}, 1\} \\ &= \{(0)^{k\,p-1}, 1\} \circ \{k\,p\,q\} \circ \{(0)^{q-1}, 1\} \\ &= \{(0)^{k\,p-1}, k\,p\,q\} \circ \{(0)^{q-1}, 1\} \\ &= \{(0)^{k\,p-1}, k\,p\,q\} \circ \{\{(0)^{q-1}, 1\}^{k\,p}\} \\ &= \{(0)^{k\,p-1}, k\,p\}, \end{aligned}$$

which depends on p and k only. For the control part this is the worst possible behaviour. During the entire period there is a delay of $k\,p$ control tokens, but the last token.

Alternatively, suppose the feedback system implements an Interpolator followed with a Decimator: $h = \mathsf{IntDec}(k\,q, k\,p)$. Here the k factor can be safely ignored since the characteristic set of coefficient of an Interpolator-Decimator network does not change with k. The closed form of s, in this case, involves two recursive sets. Since one $\mathsf{IntDec}(p, q; u)$ distributes the number of output tokens evenly over the number of input tokens of the period of h, a composition of two such systems does something similar: The p available tokens are distributed in q parts over the p entries in s. From the perspective of the control this behaviour is preferred as delays are evenly distributed over the period.

Mathematically speaking we would define s optimum for $s = \{\alpha_0 \pm \delta_0, \cdots \alpha_{k\,p-1} \pm \delta_{k\,p-1}\}$, where $\alpha \to 1$ and $\delta \to 0$, under the constraints that s meets the consistency requirement $|s|_0 = |s|_1 = k\,p$. The measure for the control lag (ϵ_i) for each token over the period of s, namely, equals

$$\epsilon_i = (i+1) - \sum_{j=0}^{i} |\alpha_i \pm \delta_i| \,; \quad \text{for } 0 \le i < k\,p$$

The average lag is minimised when the energy of s is uniformly distributed over the entire vector.

As a more practically oriented example consider an application in which a transport from an analysis process to a synthesis process uses a FIFO buffer with a finite capacity. Overloading of the buffer will break the communication. If synchronous coordination

between the synthesis and analysis process is assumed, Parks's method of an emulated bounded FIFO can be applied; see Figure 5.18 on Page 103. In case of the preferred asynchronous coordination a more elaborate scheme is required. A simple scheme to prevent overloading is the following: The analysis process labels its output data tokens with increasing numbers, the label for the next token is maintained in a counter x. The synthesis process acknowledges the proper receipt of data tokens, meaning that once every p tokens it returns the label of the most recently received data token. The analysis process maintains a "window" of W tokens. Before sending the next data token the coordination information is read (label λ). The analysis process subsequently will stop to emit data if: $x - \lambda \geq W$, i.e., when the control lags more than W samples, thus when $\max \epsilon_i > W$.

For the static case of Figure 5.22 we can derive the conditions when the analysis process will stop operating. Consider the case that has an Interpolator-Decimator feedback network, thus $s = \{(0)^{k\,p-1}, k\,p\}$, hence $\max \epsilon_i = k\,p - 1$. In this case the system will not overload the FIFO if $k\,p < W$. Alternatively consider the case that has an Decimator-Interpolator feedback network. Here we have $\max \epsilon_i = \lceil \frac{p}{q} \rceil$.

The REG link introduces schedule-dependent behaviour. The dependency can be made explicit through the specification of the control sequence c as outlined above (Figure 5.22). An implicit implementation uses the angelic-merge primitive, see Figure 5.19. In both cases, the REG link answers the question *when* events apply, *when* coordination is generated, and *when* coordination be applied. Obviously because of the schedule-dependent behaviour of the REG link, operational information may get lost. The impact of these losses with respect to the robustness of the network can only be answered in view of the functionality and the context of the network. Above we analysed a static case. A more general treatment involves a more detailed modelling of the execution patterns of the respective processes. There is an obvious correspondence with a branch of control theory that addresses systems with delay (or lag). Delay in feedback links are renown for their negative influence on the stability of a system. See for an example Section 3.5.2.

The oracle of the REG link concentrates on the indeterminate character of the network into a control stream. Oraclisation is a prerequisite for an indeterminate network to retain compositionality. In case of the REG link oracle there is a dedicated connection with the context through the control sequence c. Cooperating components may choose to connect and to gear their context-dependent behaviour to one another. Components ought to coordinate the properties of their respective control streams c. The ARC framework of Chapter 3 explicitly relies on the presence of oracles and their abstraction of context awareness: adaption and cooperation. Oracles manifest domain knowledge: the necessary context awareness of a specialised component is integrated in the respective subsystem and available for use by other systems without expert knowledge.

5.4.4 Applications

Asynchronous coordination is often found in practical systems. Many of these systems consist of a producer-consumer type of network with in between the transport network of Figure 5.23. The diagram shows two types of cooperative operational information exchange: feed forward (TxCtrl) and feedback (RxCtrl).

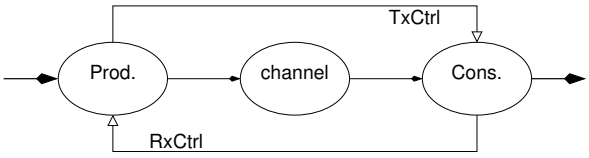


Figure 5.23: Cooperative Producer-Consumer network.

A video encoder-decoder pair is a typical example of an application that uses feedback (RxCtrl), see Example 3.3 on Page 44. Typically the implementation of the consumer (decoder) determines the display rate. More complex encoded frames require more processing resources. On the other hand, skipping frames will circumvent overloading of these resources but it is also a waste of resources since a frame skip severely degrades the overall perceived quality of the system. Typically the consumer will detect a resource overload and instruct the producer (update its context) to change a mode of operation. The system performance benefits from the cooperation, in that the quality increases and the resource utilisation decreases.

Figure 5.23 also has a feed-forward information flow (TxCtrl). This flow is useful for instance when a signal-processing filter induces a workload on a CPU system with dynamic voltage control. Given the workload information, the CPU selects an appropriate internal mode of operation; scaling its frequency and more importantly adjusting its supply voltage. The system as a whole benefits from this information by minimising waste. The workload as generated by the producer (filter) is guaranteed to be executed in time, yet it utilises just enough resources.

A related example regarding the feed forward (TxCtrl) link in Figure 5.23 concerns a cooperative video encoder combined with a flexible decoder. The encoder adapts its mode of operation based on the local context. The selected mode of operation is relevant context information for the (remote) decoder. For instance, based on the projected decoder complexity, the decoder may re-allocate its budget of local resource utilisation, for instance, it may turn from a single processor implementation to a dual processor implementation.

As a final example, consider the common internet transfer protocol TCP/IP. The buffering capacity of the transport system in between producer and consumer changes dynamically. An efficient implementation of the transfer protocol exploits the actual buffer capacity through relaxing the acknowledgement rate of correctly received data packets (events). However, an overload of the transport system yields excessive packet loss, which induces retransmits and thus it unnecessarily degrades the effective transfer rate. A consumer can derive important context information for the producer, so to optimise the system as a whole, the context information is communicated over a feedback structure (RxCtrl in Figure 5.23). The rate at which the consumer sends context information and the rate at which the producer applies the context information need not be synchronised, yielding flexible implementations of consumer and producer. As a matter of fact, working TCP/IP implementations exploit the asynchronous coordination. Looking at the evolution of TCP/IP implementations, developing a correct pro-

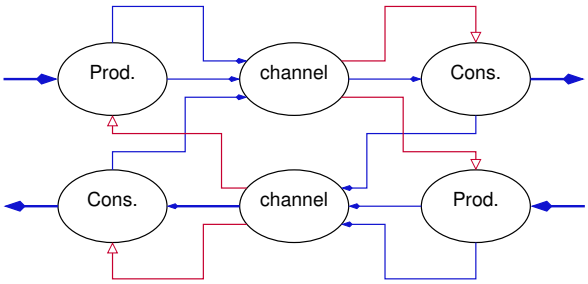


Figure 5.24: Producer consumer pair with feed forward and feedback.

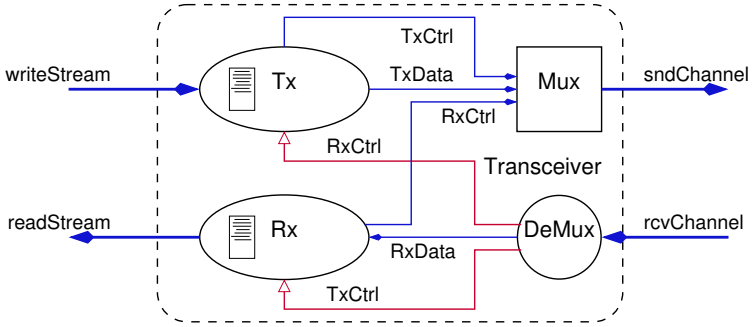


Figure 5.25: Cooperative Transceiver.

tocol is a tedious and time-consuming process. One reason is the need for exhaustive evaluations of possible environments in which the protocol has to operate adequately. However, with the use of an appropriate model of computation it is possible to isolate the context-dependent behaviour (or conditions) and subsequently to circumvent the need for extensive numerical evaluations (testing).

We have conducted an experiment to devise and verify a simplified version of the TCP/IP protocol. The basis of our experiment was a transmitter-receiver pair with feedback and feedforward *asynchronous* cooperation; Figure 5.24. A more detailed network using our CAPN constructs is the diagram of Figure 5.25. Two of these transceivers were applied, one at the producing side and one at the consuming side. The transceivers implement a protocol that can handle transport outage. The precise details of the protocol are beyond the scope of this dissertation. We applied a verification tool called SPIN [Holzmann, 1997] to verify the correctness and conditions of the devised protocol. In particular, the protocol breaks when too many transport errors occur; this is the case when the REG link fails to deliver operational information in time.

Six

Ubicom case studies

CASE studies make concrete (reify) what has been developed in the abstract. In this chapter we present a selection of cases that have been developed in the course of the Ubicom project to illustrate a number of aspects of the ARC framework. The cases predicate two import qualifications of a resulting system when applying the ARC framework: the *agility* and the *efficacy* of the system. The agility [Noble et al., 1997] of a system quantifies the ability of a system to accurately and promptly respond to the context changes. The efficacy quantifies the effectiveness of a system with respect to a measure of optimality.

In Section 6.1 we consider two cases with a cascade structure. Both cases consider a mobile video encoding system. The first case deals with a design-time evaluation, the second case describes experiments with a run-time implementation. In Section 6.2 we consider two cases with a merge structure. We consider a merge of homogeneous objects, much like the analytical case we considered in Chapter 4, and a merge of heterogeneous objects. A composition of heterogeneous objects gives rise to emergent behaviour and performance, just like the system development (composition) process itself.

We conclude this chapter with a description of the ARC interfaces (function repertoire and operation space) in the Ubicom support system.

6.1 Cascade structure

In this section we examine two implementations of a cascade filter. A design-time implementation (Section 6.1.2) and a run-time implementation (Section 6.1.3). Both implementations concentrate on the aspects of the coordination of shared resources for run-time systems. The system under observation is a mobile terminal executing a software-only video encoder, which has previously been introduced in Example 3.3 on Page 44.

The design-time implementation involves a coarse grain modelling of the system and corresponding simulation of several scenarios. It turns out that the more demanding

the application, the more the workload will be balanced equally over the components; pushing the system to its limits. The experiments show that operating the mobile terminal in a not too hostile environment makes certain components superfluous; in this particular case the channel coder becomes redundant and can be removed from the system without affecting the performance of the remaining system. This case includes an scenario-based experiment to establish the efficacy of the system when using the ARC framework. The scenario defines the context of the system: conditions and objectives.

The run-time implementation presents real-time experiments with ARC compliant components for source and channel coding. Evaluation of the experiment results demonstrates the usefulness of ARC. The system can properly operate in a broad context, and what is more: the system outperforms alternative implementations when applied in a resource scarce environment. This case includes an experiment to establish the agility of the resulting system and to establish the overhead for using the ARC framework.

6.1.1 System description

We consider an application that distributes and collects multiple viewpoints of a scene. A typical scenario involves multiple mobile terminals each equipped with a camera and video capture equipment. Users may retrieve and view video streams from each of the terminals. The user is offered a choice of life-video streams with per stream a choice of quality versus power dissipation options. The user may choose to view good quality images for a short while or to view perceptively lower quality images for some longer while. Perceptive quality is a single parameter that is related to frame rate, distortion, and resolution.

The system is composed of a set of building blocks (components) with a function repertoire as can be expected in any system for visual mobile augmented reality: source coding, channel coding, and transmission. Composing the functional system is straightforward in this application. It is a cascade in which workload of each of the components is routed via a neighbouring component. For the sake of illustration we choose a very basic mobile terminal architecture with a single CPU, see the diagram of Figure 6.1. The subsequent mapping of functionality to resources is therefore trivial; all computations but the analogue transmit part of the transceiver component are mapped to a single CPU system. The manager and viewer (not shown) functionality are mapped to individual mobile terminals.

The diagram of Figure 6.1 also specifies the ARC interfaces: the operation space parameters and function repertoire. The function repertoire is implicitly defined in the diagram. We notice the usual classes of operation space parameters: distortion, capacity, and resource utilisation. For the design-time evaluation this number of parameters is adequate. For the run-time implementation, however, we need at least one additional resource utilisation parameter, namely latency. Next to power dissipation, latency is an exhaustible resource in real-time signal processing.

The applied models do not take into account the negative effects of time-sharing. As an example, we assume in both case studies the instantaneous availability of CPU cycles. However, it is a well known fact that sharing of a CPU induces overhead; although the sum of the workloads is guaranteed never to exceed the CPU capacity, timesharing may

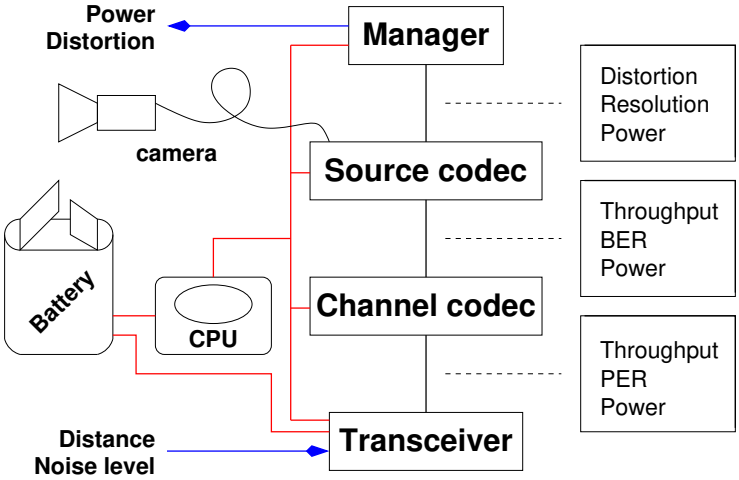


Figure 6.1: View point sharing, the structural model and mapping. The ARC interface parameters are specified on the right-hand side.

prevent timely availability of the scheduled workload. Frequent switching of tasks in real-time operating systems typically, degrades the effective CPU capacity significantly. Although neglecting interference caused by time-sharing is an over-simplification of a practical design it does not break the ARC framework. If necessary such a interference can be modelled explicitly. For instance by partitioning each of the component in a sending (encoding) and a receiving (decoding) part. The receiving part may *sense* (monitor) the actual access to the CPU cycle budget. This information can be referred to the sending part and incorporated in its context. This context information may trigger the sender (encoder) to actualise its resource utilisation model.

6.1.2 Mobile video communication: design time

The mobile video coding system is built of basic models that capture the significant behaviour, performance, and resource utilisation of a component. We have simulated a system as given in Figure 6.1 using a generic mathematical tool: Mathematica™. Calibration of the models is important when evaluating the system quantitatively. The experiments have been conducted using calibration data from [van Dijk and van Reeuwijk, 1999]. Note though, that the system model and the coordination mechanism among components does not change if we were to replace a component with a more suitable, differently calibrated, component.

Table 6.1 presents the parameters for each of the components, both the interface parameters between components and the internal state parameters of components are specified. Not shown, but important, is that the power resource utilisation parameter is implemented as a tuple: ⟨power, cycles⟩. In which the power entry represents the total power dissipation of the employed subsystem, and the cycles entry is the aggregation

Table 6.1: Parameter definition for components interfaces and their internal state.

User	*schedule*	;	*intentions and agenda*
Distortion	; subjective quality for the given application		
Power	; total power dissipation		
Manager	F_{max}	;	*fixed maximum desirable frame rate*
Distortion	; discarded variance after transfer		
Frame rate	; number of frames per second		
Power	; video transfer power dissipation and CPU utilisation		
Source codec	F_{res}	;	*frame resolution: fixed to* CIF
	ℓ	;	*encoded variance*
	c	;	*coding rate*
	(F_r, gop)	;	*source characterisation: rate and grouping*
Throughput	; bits per second		
BER	; bit error rate		
Power	; link transfer power dissipation and CPU utilisation		
Channel codec	n	;	*packet size*
	k	;	*source bits per packet*
Throughput	; bit rate		
Pe	; error probability		
Power	; channel transfer power dissipation CPU utilisation		
Transceiver	N	;	*number of* OFDM *sub carriers*
	M	;	*modulation scheme, bits/symbol*
	f_s, f_{os}	;	*AD/DC sample rate and oversampling factor*
	P_{tx}	;	*transmit power*
Channel	N_i	;	*interference noise level*
	d	;	*distance between mobiles*
CPU	f_{th}	;	*clock frequency threshold*
	Vdd	;	*supply voltage*

of the CPU cycle workload induced by the employed subsystem. This way, coordinating the cycle budget of the CPU-system is effectively distributed over all components.

In the remainder of this section we present some internal details of the components. Each component has been modelled with workload generation and performance indication functions. The workload generation functions correspond to the consistency constraints with respect to the offer from the employed services, see Equation (4.4) on Page 64. The performance evaluation functions correspond to the consistency constraints with respect to the request from the consulting client, see Equation (4.5) on Page 64. ARC operation spaces require pruning; only a subset of the set of Pareto points is included. Therefore, optimisations must be carried out. When appropriate, the optimisation criteria are made explicit.

Table 6.2: CPU system model

CPU (and Memory) system
Performance $\text{CPUPower}(cycles) \propto \text{Vdd} \cdot cycles + \text{Vdd}(cycles)$
Optimisation $\text{Vdd}(cycles) = \text{if } (cycles > f_{th}) \text{ then high else low}$

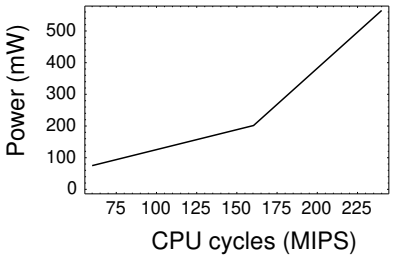


Figure 6.2: Operation space of the CPU system.

CPU system

The CPU system consumes energy while delivering performance. Modern processors support dynamic voltage control (DVC), which make them context aware [Pouwelse et al., 2000]; offering just enough performance while consuming minimum energy. We incorporate a simplified version of DVC, instead of a continuous tradeoff we use a piecewise linear model with two modes of operation: a low power low performance mode and a high power mode with corresponding high performance. The CPU system is a resource only system, so the model only needs a performance indication. In this case study we silently ignore the memory system. Table 6.2 provides the details of the model. Figure 6.2 presents the corresponding performance indication model, which corresponds to the operation space of the CPU system. The proper coordination of the CPU system is a joint responsibility of all components that consume CPU cycles. Compatibility checks must prevent an overload of the CPU system. The proper operation of the CPU system is the responsibility of the CPU system component designer, such as switching of the supply voltage. Whether the supply voltage is defined through contracts or through other means is left unspecified.

Transceiver system

The transceiver system incorporates a model of the underlying physical channel. The channel is modelled with two parameters, interference noise (N_i) and mobile-to-mobile distance (d). The transceiver implements orthogonal frequency division modulation (OFDM) coding with quadrature amplitude modulation (QAM). The parameters of the

Table 6.3: Transceiver system model

Transceiver
Work load $\text{Bandwidth}(f_s) = f_s / f_{os}$ $\text{Signal}(P_{tx}) = P_{tx}$ $\text{CPUcycles}(N, M, f_s) \propto (N \cdot {}^2\log(N) + M^2)\ \dfrac{f_s}{1+N}$
Performance $\text{Throughput}(N, M, f_s) \propto \dfrac{f_s \cdot N \cdot M}{1+N}$ $\text{Pe}(P_{tx}, N, M, d, f_s) \propto \dfrac{N-1}{2}\ \text{Erfc}\left(A \cdot N_i \sqrt{\dfrac{(1+n)P_{tx} \cdot {}^{10}\log(M)}{(M-1) \cdot d^{\alpha} \cdot f_s \cdot M \cdot N}}\right)$ $\text{Power}(P_{tx}, N, M, f_s) \propto P_{tx} + \text{CPUPower}(\text{CPUcyles}(N, M, f_s))$
Optimisation $\min_{P_{tx}, N, M, f_s} \text{Power}(P_{tx}, N, M, f_s)$; given (Throughput, Pe)

coding and modulation are subject to optimisation. Likewise, the parameters of selecting a proper transmit power and AD/DC sample rate are subject to optimisation; see also Table 6.1. The transceiver applies fixed oversampling for the AD/DC converters and adds an also fixed cyclic prefix to OFDM coded frames.

The channel model assumes a cellular network with a 400 m cell radius. We apply the path-loss formula Equation (6.1). For a network with a 17 GHz carrier frequency, the parameters A and α were empirically determined in [Bohdanowicz, 2000] at $A = -66.6$ dB and $\alpha = 1.70$.

$$A \cdot d^{-\alpha} \tag{6.1}$$

The interference noise (N_i) has been set to the equivalence of 18 maximum interfering neighbouring cells located at 3 times the cell radius. The interference noise is about 70 times the system background noise ($N_o = k\ T_{sys}\ BW$, where T_{sys} is the system temperature of 300 K , BW the (frequency) bandwidth and k Bolzmann's constant ($1.3807 \cdot 10^{-23}$ J K^{-1}).

The transceiver system generates a workload for two resources, the physical channel and the CPU system. With respect to the physical channel, the occupied bandwidth and emitted transmit power are the important metrics. Other terminals regard this as additional (dynamic) noise. The induced workload on the CPU, for doing computations, yields a power dissipation figure. The operation space of the transceiver is in terms of throughput, error probability, and power usage, see Table 6.1. The optimisation routine finds the best possible solution in terms of power dissipation for a requested throughput and error probability. Table 6.3 shows the details.

The offered error probability, Pe, uses a well known trend function, see e.g., [Lee and Messerschmitt, 1994]. Figure 6.3 illustrates Pe, with a normalised energy per bit

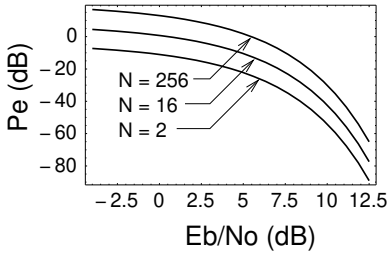


Figure 6.3: Illustration of transceiver design space.

(E_b/N_o). The diagram shows that for a given modulation scheme, the resulting error probability improves with the assignment of more energy per bit. The curves discriminate between the number of sub carriers used in OFDM coding. With the increase of the number of carriers, the error probability increases as well, assuming a given energy per bit.

Channel coding

Channel coding trades off throughput with bit error rate while transferring data over an unreliable channel. The channel coding system relies on the transceiver for a specification of the channel model, i.e., the operation space offered by the transceiver. Channel coding is a CPU intensive component.

Inducing a workload on the transceiver system must match an entry in the operation space exposure, i.e., ⟨throughput, error probability, power⟩. Therefore it suffices to present the workload induced on the CPU. The workload on the transceiver system is implicitly incorporated in the performance indication functions. For the performance model we use a theoretic approximation model based on the Shannon capacity of a binary symmetric channel Equation (6.2). Here p is the crossover error probability of the channel and c its capacity. Capacity is a strict decreasing function $(1 \to 0)$ on the interval $p \in [0, 0.5\rangle$.

$$c = 1 + (1-p)\,^2\log(1-p) + p\,^2\log p \tag{6.2}$$

Given the capacity of a channel $c = k/n$, find the corresponding p. Let $p = p'$ solve Equation (6.2) for $c = k/n$. Let p_e be the channel error probability. We introduce $\mu = n \cdot p_e$, and $t = \lfloor n \cdot p' \rfloor$. The bit error rate (BER) of the resulting link is approximated by substituting μ, t in the cumulative Poisson distribution of Equation (6.3).

$$\begin{aligned} \mathrm{CDF}_{\mathrm{Poisson}} &= \sum_{x=t}^{\infty} \exp(-\mu)\,\frac{\mu^x}{x!} \\ &= \exp(-\mu)\left(\exp(-\mu) - \frac{\exp\mu \cdot \mathrm{Gamma}(t,\mu)}{\mathrm{Gamma}(t)}\right) \end{aligned} \tag{6.3}$$

Table 6.4: Channel codec system model, see also Equation (6.2) and Equation (6.3).

Channel coding system
Work load $\text{CPUcycles}(th_{put}, p_e, n, k) \propto th_{put} \dfrac{(n-k)(1+50(p_e)^{3/10})}{n}$
Performance $\text{Throughput}(n, k, th_{put}) = \dfrac{k}{n} th_{put}$ $\text{BER}(n, k, p_e) = \text{CDF}_{\text{Poisson}}((\mu = n \cdot p_e), t)$ $\text{Power}(n, k, th_{put}, p_e, P) \propto P + \text{CPUPower}(\text{CPUcycles}(th_{put}, p_e, n, k))$
Optimisation $\min(Power/Throughput)$; given (BER)

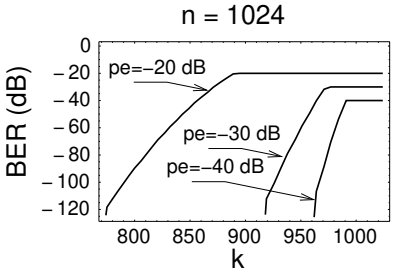


Figure 6.4: Illustration of Channel codec design space.

An optimisation routine filters out unattractive operation points from the perspective of minimising the energy per bit sent. Filtering out is based on the requested BER ranges. More details can be found in Table 6.4.

Decreasing the number of source bits k in a packet of size n yields an improvement of the link quality. Figure 6.4 illustrates the typical trend functions in various context, i.e., the channel error probability (pe) as offered by the transceiver.

Source coding system

The source coding component applies a progressive coding technique. The encoded variance, a perceptive quality related measure, thus increases with the number of bits sent. Due to link errors the subsequent transferred amount of variance is less than the encoded variance. The perceived distortion is modelled as the ratio between the transferred variance and the variance of the source [van der Schaaf and Lagendijk, 2000].

The encoded variance is proportional to the logarithm of a normalised block length

Table 6.5: Source codec system model, see also Equation (6.4).

Source coding system
Work load
$\text{Throughput}(\ell, c_r, f_r) = \ell \dfrac{f_r \cdot \text{CIF}_{res.}}{c_r}$
$\text{CPUcycles}(\ell, c_r, gop, f_r) \propto \dfrac{\text{Throughput}(\ell, c_r, f_r) \cdot c_r^2}{1 + {}^{64}\log gop}$
Performance
$\text{Distortion}(\ell, c_r, gop, f_r, p_e) = (1 - \text{encVar}(\ell, \text{m}(f_r)))\ (1 - p_e^{1/5})^{\frac{f_r \cdot \ell}{c_r \cdot gop}}$
$\text{Framerate}(f_r) = f_r$
$\text{Power}(\ell, c_r, gop, f_r, P) \propto P + \text{CPUPower}(\text{CPUcycles}(\ell, c_r, gop, f_r))$
Optimisation
$\min\ (\text{Power}/(1 - \text{Distortion}))\ ;\quad \text{given (Framerate)}$

ℓ:

$$\text{encVar} = \ln\left(\frac{\ell}{m} + 1\right) \tag{6.4}$$

in which m characterises the source. Typical values for m are: $m = 2^{(-16)}$ for static still images, $m = 2^{(-8)}$ for natural images, $m = 2^{(-3)}$ mimics a video stream and $m = 1$ the notorious MTV video clip.

The mapping of the normalised block length ℓ to the effective transferred variance and the corresponding throughput, is by means of a group of pictures (gop) variable and a coding rate ($0 < c_r \leq 1$). The maximum absolute block length depends on the (fixed) frame resolution parameter. Optimisations finally, are based on minimising the energy per quality setting while considering the requested range for frame rates. See Table 6.5 for more details.

The distribution of variance over an progressive coded bit stream is logarithmic, see Equation (6.4). Figure 6.5 illustrates the (normalised) curves for a number of sources. In this case study the source characteristics (m) have been coupled with the frame rate. Hence $f_r = 1$ corresponds to a still image and $f_r = 25$ corresponds to an MTV video clip.

Video manager

The video manager (broker) maps the distortion (δ) and frame rate (f_r) offered by the source codec to a perceptive quality measure. The manager prefers low distortion and high frame rates. Frame rates are valued relative to the maximum desired frame rate of $F_{max} = 25$ fps. The optimisation process selects the lowest possible energy consumption per requested quality measure. Table 6.6 gives some more details.

Mapping of distortion and frame rate tuples to a scalar quality number is illustrated in Figure 6.6. The exponential character of the formula for the quality figure complies

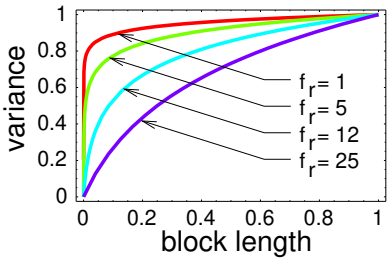


Figure 6.5: Illustration of Source codec design space.

Table 6.6: Video manager system model

Manager system
Work load CPUcycles( ) = 0
Performance $\text{Quality}(\delta, f_r) = (1-\delta)\left(\frac{f_r}{F_{max}}\right)^{\sqrt{3}}$ Power(P) = P + CPUPower(CPUcycles(0))
Optimisation min (Power/Quality) ; given (Quality)

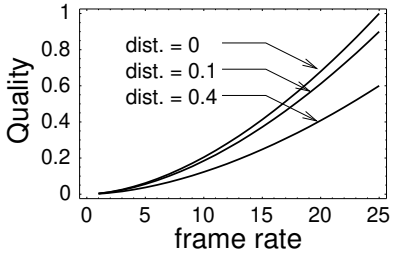


Figure 6.6: Illustration of Video broker design space.

with intuition. The selection of the precise value of the exponent ($\sqrt{3}$) is due to illustration purposes. A system with either a shallow or harsh video manager are not particularly interesting. With a shallow video manager, the load on the rest of the components is always relative low, whereas with a harsh video manager the load on the rest of the components is always high.

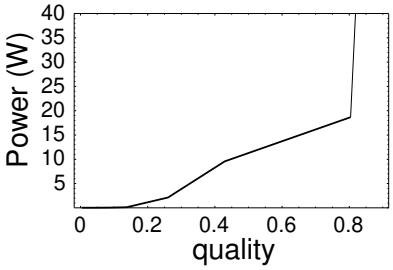


Figure 6.7: Illustration of the manager operation space ($d = 100$ m).

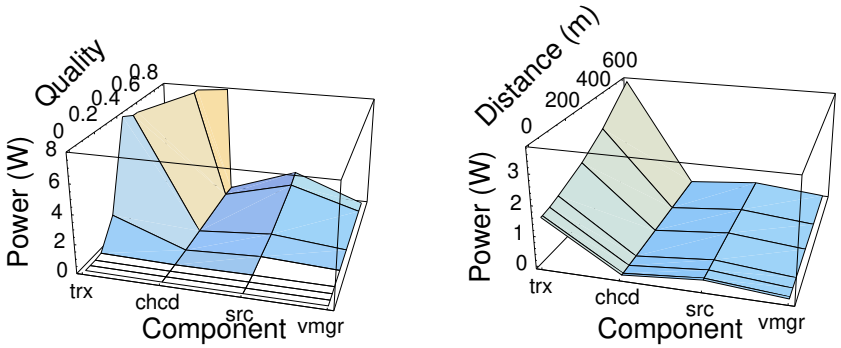
System analysis

We have evaluated the presented model for a range of scenarios. One scenario is to fix the channel conditions and to issue requests from the user to the system that covers the whole range of quality values. Figure 6.7 illustrates the operation space with fixed channel conditions; a mobile-to-mobile distance of 100 m. Obviously, the proposed implementation of the system is real power hungry, a feasible system could never reach a quality better than $Q = 0.8$.

The evaluated scenarios cover channel conditions in which the mobile-to-mobile distance ranges from 1 to 600 meters and they cover user objectives in which the requested system quality level ranges from 0 to 0.9. Unfortunately, delivering high quality at long distance is prohibitively costly (not shown, but the system dissipates in this case up to 400 W).

Consider two scenarios, one with fixed mobile-to-mobile distance and varying quality level, and one with fixed quality level and varying mobile-to-mobile distance. The diagram of Figure 6.8(a) shows the distribution of the power budget over the respective system components when conditions are fixed ($d = 100$ m). Obviously, the transceiver power grows exponentially with the quality level. The source coder requires an amount of energy that is proportional to the system quality level. Channel coding is superfluous up till the point where the system quality level reaches $Q = 0.4$, for higher qualities, channel coding is invoked.

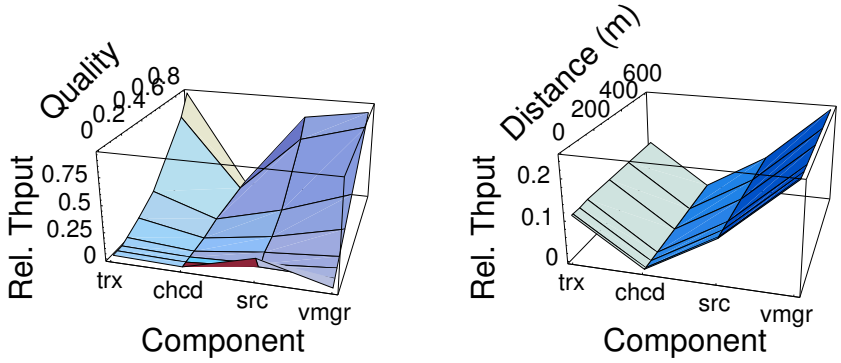
The diagram of Figure 6.8(b) shows the distribution of the power budget over the respective system components when the objectives are fixed ($Q = 0.25$). In this particular scenario the mobile-to-mobile distance does not induce changes in the mode of operation of either of the components. Closer, off line, analysis shows that only the transmit power of the transceiver is adjusted when the observed distance increases. On the other hand, when we fix the distance, every component will operate in every possible mode of operation when the objective quality level varies from $Q = 0$ to $Q = 0.8$. The above conclusions are backed up by Figure 6.9(a) and 6.9(b). These figures show the relative "throughput" per component. Relative throughput is a component specific metric. The transceiver defines relative throughput as the ratio of the actual bit rate and the maximal supported bit rate. The channel codec defines relative throughput as the redundant portion of the transport stream and the source coder defines relative through-



(a) $d = 100$ m. (b) $Q = 0.25$.

`trx`: transceiver; `chcd`: channel codec; `src`: source codec; `vmgr`: video manager

Figure 6.8: Power budget distribution per component.



(a) $d = 100$ m. (b) $Q = 0.25$.

`trx`: transceiver; `chcd`: channel codec; `src`: source codec; `vmgr`: video manager

Figure 6.9: Relative throughput per component.

put as the normalised encoded variance (ratio of the encoded variance and the variance of the source).

Reaching the bounds of a posed request may indicate a possible bottleneck that can hamper us in finding the system-wide optimum. Bottlenecks are simply removed by relaxing the bounds of the request. In the scenarios that we considered here this is not necessary since we are at the extreme end of the system-wide objective of perceived quality levels.

Discussion

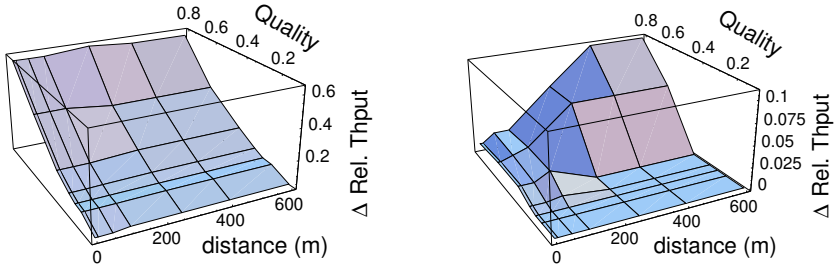
The experiments with the system model include an exhaustive design space exploration, which combines all possible options of every component in the system into a single huge optimisation space. Even in a small scale case as the one presented here this introduces all kind of scalability issues, long computation times, and extensive memory usage. More details can be found in [van Dijk et al., 2000b]. The ARC protocol of doing request-offer-select effectively navigates through the same huge design space but does not run into scalability issues. To validate this conjecture we have evaluated all operation spaces between the components of the system. It turns out that the eventual selected operation point, given the system context, never exceeds the requested range. Hence both methods find the same, optimum, mode of operation for each component. Obviously, ARC is several order of magnitude more efficient. In this particular case, exhaustive design space exploration requires the evaluation of $\mathcal{O}(10^{12})$ operation points[1], whereas the ARC protocol requires $\mathcal{O}(10^2)$ In fact, we rely on the monotonicity of subsequent operation spaces.

The graphical interpretation of operation spaces and their gathering (e.g. Figure 6.9(a)) prove extremely useful in interpreting the results of the analysis. As an example, we consider the agility of the system. The agility quantifies the rate of change from one mode of operation to another. The necessary insight can be derived through a reordering of experimental results. Assume a scenario in which mobile terminals move with a speed of 1.4 m/s (walking speed). Figures 6.10(a) and 6.10(b) give the rate of change of the previously mentioned relative throughput of the respective components. The source coder and transceiver (not shown) have comparable behaviour. Their agility depends on the requested quality level of the system, but is independent form the (initial) distance between transceivers. In order to keep up with a moving terminal at walking speed, moving away from the source, the source coder must adapts its relative throughput with an absolute value of 0.4 when the overall system quality level is set at $Q = 0.6$. The channel codec, however, has less smooth behaviour with respect to agility: at large distance and high quality the agility is an order of magnitude higher than at small distances or low quality.

6.1.3 Mobile video communication: run-time implementation

A run-time analysis of the mobile video communication system of Section 6.1.1 yields quantitative performance numbers. The experiments address issues of the efficacy and the agility of such a system. The run-time implementation described here concentrates on the encoding part of the video codec and the protocol components. We assume, without loss of generality, that the applied transceiver component is non-adaptive, yet the time-varying aspect of the channel model is maintained. The video encoder and protocol components implement the ARC framework for doing QoS negotiations. In this section we highlight the principle details of their implementation and discuss a number of conducted experiments.

[1] 14 free parameters, with a moderate 8 possible settings each yields $8^{14} \approx \left(2^{10}\right)^4$ operation points.



(a) Source coder. (b) Channel coder.

Figure 6.10: Relative throughput rate of change.

Table 6.7: QoS parameters (descending priority).

Parameter	Description
latency (max)	The time required to transmit a single bit.
bit-error rate (max)	The net bit-error probability after FEC and ARQ.
CPU usage (max)	The allowed share of CPU capacity for protocols and transmission processes.
throughput (min)	The minimum net throughput.

The QoS parameters at the interface between the video encoder and protocols components are given in Table 6.7. For a detailed description of the design of this interface we refer to [van der Schaaf et al., 2001]. The interdependency of parameters is an explicit account of the mutual interpretation of operation space parameters. The fact that the parameters are prioritised is due to a run-time optimisation. Moreover each parameter is partially ordered; both components have a proper understanding for each parameter of which direction is worse (higher bit-error rate) and which direction is better (higher throughput). The system operates in a varying context with varying objectives and varying conditions, which implies there must be a protocol for handling out-of-contract operations. Renegotiation is covered by the ARC framework. Yet when the mode of operation violates the current contract, parties must decide whether or not and how to continue operation during a renegotiation phase. In this particular case there is a generic preferred way of handling out-of-contract operations. First sacrifice (lower) the throughput, then sacrifice (lower) the CPU usage, then (increase) the bit-error rate, and so on. Obviously this choice of predefined, inter and intra parameter, ordering is implementation dependent. A more flexible method is to introduce additional ARC interface parameters that describe how to continue when operating out-of-contract. Although more flexible, some overhead is unavoidable. In this particular case such added flexibility is regarded an overkill.

Video encoder model

The video codec implements a flexible H.263 encoder/decoder pair. Internally, there are numerous video encoding parameters that can be tuned. In the context of ARC, we create an abstract description of the behaviour of the encoder, extending the work of [Girod and Farber, 1999; Lan and Tewfik, 1998; Stuhlmüller et al., 2000]. The dominant decision variables considered here are the frame-skip N_{fs}, the maximal motion vector length L_{mv}, the intra-block refresh rate β, and the target bit-rate R. The variable user objectives are observed through the characteristics of the video sequence, which are gathered in a model of the temporal predictability and the (average) amount of variance σ_o^2.

The control model focuses on *a priori* estimation of the rate-distortion behaviour as a function of R, L_{mv}, N_{fs}, the source characteristics, and the choice whether or not to use motion compensation. The prediction gain G is a principal parameter of the rate-distortion model. The prediction gain expresses the similarity between the actual version of a signal and its predicted version. Let σ_x^2 be the actual variance of a signal x and $\sigma_{\Delta x}^2$ be the variance of the error of the predicted version of the signal, then G is simply defined as:

$$G = \frac{\sigma_x^2}{\sigma_{\Delta x}^2}$$

A model of G depends on the internal parameters of the encoder as well as source model parameters. Let G_o be the prediction gain without motion compensation and without frame skips ($N_{\text{fs}} = 0$). Similarly, let G_{mv} be the prediction gain with optimum motion compensation, still without frame skips. In addition we have two other parameters that model the source characteristics: the motion coherence, N_{mc}, and the average motion vector length ℓ_{mv}. Obviously, the actual value of G ranges from G_o to G_{mv} ($G_o \leq G_{\text{mv}}$) if $N_{\text{fs}} = 0$. When $N_{\text{fs}} > 0$ a correction factor is required; G increases with N_{mc} but decreases with N_{fs}. Apart from this correction factor, G scales with ℓ_{mv} and N_{fs}. The model of G, given in Equation (6.5), falls from G_{mv} to G_o with the increase of ℓ_{mv} or N_{fs}.

$$G = 1 + \left(G_{\text{mv}} - (G_{\text{mv}} - G_o) \exp \frac{-R_{\text{mv}}}{\ell_{\text{mv}}(N_{\text{fs}} + 1)} \right) \exp \frac{-N_{\text{fs}}}{N_{\text{mc}}} \tag{6.5}$$

The average amount of variance that has to be encoded depends on G, σ_o and the intra-block refresh rate β. This latter parameter limits cumulation of errors, caused by erroneous transmission. The encoded variance σ_d^2 is given as

$$\sigma_d^2 = \beta \cdot \sigma_o^2 + (1 - \beta)\frac{\sigma_o^2}{G} \tag{6.6}$$

The distortion after transmission and decoding is quantified as a peak signal-to-noise ratio (PSNR). Several noise sources contribute to the PSNR value. The encoding process adds quantisation noise D_q, which is inverse proportional to the bitrate R. A second source of noise, D_t is the due to transmit errors. D_t is proportional with β and

depends on the BER of the channel through $p_e = 1/\text{BER}$. A final source of noise, D_s, is due to skipping of frames. These noise contributions are given as follows:

$$\begin{aligned} D_q &= \frac{\theta}{R} \\ D_t &= \left(1 - (1 - p_e)^{R \cdot N_{\text{fs}}} \sigma_o^2\right) \sum_{t=0}^{t=1/\beta} \frac{1 - \beta \cdot t}{1 + \gamma \cdot t} \\ D_s &= \sum_{f=1}^{N_{\text{fs}}} G_o \exp \frac{1 - f}{N_{\text{mc}}} \end{aligned} \tag{6.7}$$

The parameter θ is an empirical model parameter, which established through calibration. The parameter γ is the so-called *leakage* [Stuhlmüller et al., 2000]. D_t is proportional with β when $\gamma = 0$.

The resulting PSNR (in dB) normalises the collected noise contribution as in Equation (6.8).

$$\text{PSNR} = 10\,{}^{10}\log \frac{255^2}{D_q + D_t + D_s} \tag{6.8}$$

The distortion model, models the throughput and capacity aspects of the video coder in its context. In addition, models for resource utilisation of CPU and latency have been developed. These resource utilisation models are inverse proportional with the frame-skip factor and increase quadratically with L_{mv}. Obviously there is a tradeoff between the achieved distortion, CPU utilisation, and latency. The encoder offers the manager an operation space with non-inferior points.

Protocols model

The communication protocols employ a simple transceiver that offers a time division multiple access (TDMA) scheme with fixed sized transmission/receive slots to access the physical channel. Due to interference, fading, and other factors the data transmitted over the radio channel is subject to errors. The protocols implement two methods to counter the high bit-error rate (BER) of the physical channel: forward error correction (FEC) and automatic retransmit requests (ARQ).

Note that, contrary to many implementations, FEC is implemented in software. We use a Reed-Solomon protection scheme with four different code rates $c = k/n$: 256/256, 224/256, 192/256, and 128/256. Where n is the unit packet length and k the effective unit message length; leaving $n - k$ units for error handling. The code rate determines the maximum effective throughput that can be offered. Simple models have been devised for modelling the resource utilisation of CPU and latency and to establish the effective BER after transmission. The resource utilisation models depend only on the code rate c, the BER model also takes into account the packet error P_{err} of the physical channel. A series of off-line tests determine the computational complexity and effective BER of the four code rates using white Gaussian additive noise. These results have been collected in a lookup table, which is consulted during run-time execution.

Packets that must be delivered reliably (i.e., without any error) are extended with a 32-bit checksum. When the receiver observes a checksum failure, it returns a retransmit request to the sender, who will in turn re-send the packet. ARQ increases latency (L) and reduces the effective throughput (T) that can be obtained. We use the following model to quantify the efficiency loss due to retransmits:

$$T_{\text{ARQ}} = \frac{T_{\text{FEC}}}{1 + P_{\text{err}}} \tag{6.9}$$

$$L_{\text{ARQ}} = (1 + 2P_{\text{err}})\, L_{\text{FEC}} \tag{6.10}$$

where P_{err} is the probability that a packet is corrupted.

By combining the lookup tables for FEC and the ARQ model, the protocol layer can quickly evaluate the settings (code rate + enable/disable ARQ), given the current channel conditions and contract with the video encoder. When requested an offer, it prunes any inferior points out of eight alternatives.

Experimental Evaluation

The experiment set up is as follows. A pre-recorded video stream (`carphone`) is encoded at a mobile terminal, transmitted over a (simulated) wireless link and decoded at a base station. Latency requirements are such that *live* viewing is possible ($L <$ 0.5 s). The transceiver has fixed settings: constant transmission power and constant modulation schemes during the length of the experiment. However, interference at the physical channel will occur, causing an increased bit error rate at the radio channel.

We present here three experiments. For each experiments we assume a user who requests the best possible quality within 100% CPU budget, moreover we apply an video encoder that does not exploit any changes in the characteristics of the source. Afterwards we describe an experiment that does adapts to these changes.

i. *Steady run*. There is constant interference on the raw wireless channel. Therefore neither of the components adapts during this experiment. The video encoder does not even adapt to changing characteristics of the incoming video source. We conducted this experiment for three possible channel states: 1) a bad channel, BER = $2 \cdot 10^{-2}$, 2) a medium channel, BER = 10^{-2}, and 3) a good channel, BER = 10^{-4}. The raw channel bitrate is kept at 1.8 10^4 bit/s.

ii. *Frozen run*. After 20 seconds from the start of the experiment the initial *medium* channel changes to a *bad* channel. The throughput of the raw channel is maintained at 1.8 10^4 bit/s. At $t = 47.5$ s the channel changes to a *good* state. The protocols layer adapts instantaneously to the changed raw channel conditions. Initially, it maintains the contracted BER at the link to the video encoder but it has to sacrifice throughput at the link. The video encoder establishes an initial contract assuming a (worst-case) BER of $2 \cdot 10^{-2}$ right after the start of the experiments. For the remainder of the experiment, all internal settings (and contracts) are frozen. To maintain the agreed CPU budget and real-time objectives, some frames will be skipped, which decreases the delivered quality.

iii. *Adaptive run*. The physical channel and protocols layer behave as in the frozen run above. This time, however, the video encoder initiates QoS negotiations and adapts to the changed conditions. Like in the frozen run the net effect is that the video encoder maintains the agreed real-time and CPU-budget constraints.

Figure 6.11 presents the results of the experiments. The top diagram shows the BER of the raw channel for the steady cases of Experiment i and the time varying cases of Experiments ii and iii. The diagram in the middle shows the effects on the throughput delivered by the protocols to the video encoder (Experiments i and iii). The bottom diagram has four curves, one for each of the three "steady" runs and one for the adaptive run. The results of the frozen run closely follow the steady run with the "bad" channel conditions, and is left out for clarity. The curves plot the quality (PSNR) per received frame. As can be expected the steady run with "good channel" conditions has the highest quality. Quality variations over time are due to variations of the input source characteristics. Observe that the curve for the adaptive run switches between the three steady curves (medium→bad→good) when the channel conditions change. This demonstrates that ARC-based negotiations succeed in selecting appropriate settings of the video coder and outperform a coder that assumes worst-case conditions (Experiment ii). For the adaptive run, channel conditions change significantly at $t = 47.5$ s, indicated a in Figure 6.11. The encoder reacts within a couple of encoded frames, and after a few dozen frames the encoder operates at the expected level; the *agility* of the system is quite acceptable.

Compared to the steady run, with *good* channel conditions, the average quality of the frozen run is 1.97 dB lower. The average quality of adaptive run is only 0.91 dB less than the steady run. In the adaptive run, 864 operation points were evaluated in three rounds of negotiations. The total time needed for these quality of service negotiations is 80 ms. It is instructive to present the internal settings of the respective components for each of the experiments. Table 6.8 shows the results. The values shown for the frozen and adaptive run are averages, because the parameters are changing over time.

We conducted a fourth experiment

iv. *Source adaptation*. The physical channel operates in the *bad* state as in the steady run (Experiment i). The users requests the best possible quality as before but this time with a limited CPU budget of 30%. The video encoder observes variations in the source characteristics.

The system-wide quality (PSNR) depends amongst others on the source characteristics. The bottom diagram of Figure 6.11 suggest a drastic change at $t = 34$ s, marked b. In the first three experiments any variations in the source characteristics were neglected. The video encoder, however, can adapt to these changes. When doing so this results, on average, in an improved quality of 1.1 dB. Note that this result is under tight objectives of limited resource utilisation; the user can trade quality for resources.

Discussion

The experiments demonstrate the agility of the implementation. The ARC framework improves the overall performance in cases where the channel conditions or video

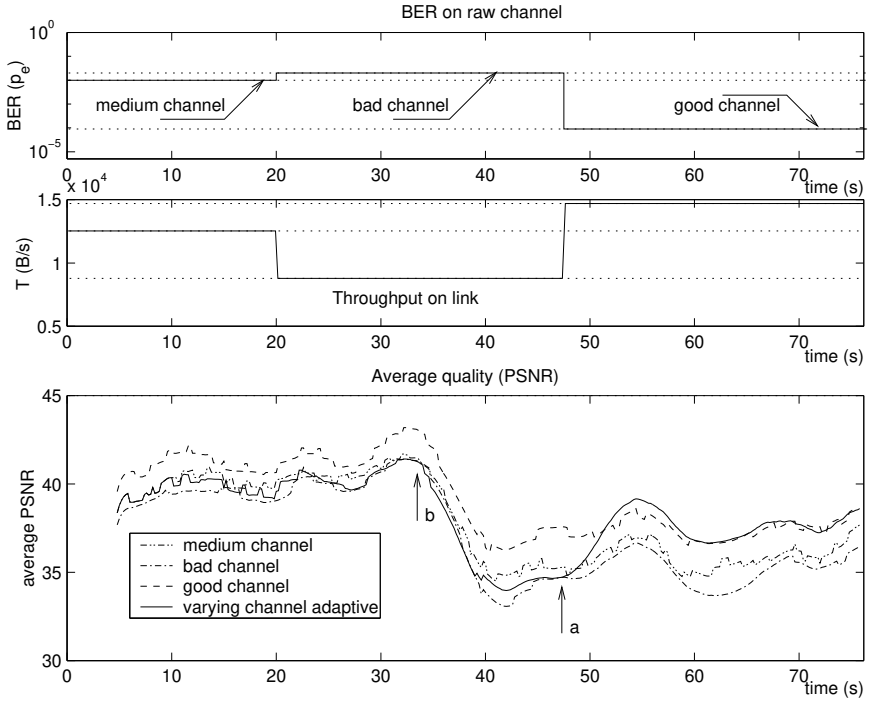


Figure 6.11: Experimental results.

source characteristics fluctuate. Moreover the ARC framework keeps the resource usage within bounds, even when the channel status is changing; thus efficiently applying the available resources.

The performance models developed in the context of the component are specific for the applied schemes in the respective components, H.263 and FEC/ARQ respectively. They manifest the functionality, behaviour, and performance of the respective components. Application of a different scheme for a component would require a different model, yet seamless incorporation in the ARC framework is guaranteed. Note that the models presented here have been developed by the domain expert, this is the preferred situation.

The robustness of the set-up may be questioned when the context changes frequently. In such situations the optimiser will lag behind, and persists in making the wrong decisions. Here, this problem is circumvented by relaxing the contract margins. This way components can perform internal adaptations at the expense of being suboptimal.

Table 6.8: Parameter settings.

	steady			frozen	adaptive
	medium	bad	good		
L_{mv}	12.00	9.00	12.00	9.00	14.30
R (bpp)	0.67	0.47	0.79	0.47	0.51
N_{fs}	1.15	1.03	1.07	1.01	1.04
Video CPU-budget	50%	45%	56%	52%	86%
FEC	192/256	128/256	256/256	146/256	159/256
ARQ	0	0	0	0	0
Proto CPU-budget	6%	14%	2%	8%	7%

6.2 Parallel structure

In this section we evaluate the cooperation of components in a merge structure. Merging of objects (or functions or components) is a frequently applied method of abstraction. A typical example from the field of communication systems is the Always Best Connected (ABC) concept [Frodigh et al., 2001]. ABC offers a single, functional, interface for data exchange between communication partners. Depending the context, a specific physical transfer system is selected. A typical example of ABC in the field wireless transmission combines Irda for back-to-back communications, WiFi for local area communications, and GSM as a fall-back option.

ABC is an example of a merger of homogeneous components. Although the physical components are distinct, ABC concentrates on their functionality to establish a physical point-to- point link so as to facilitate the internet protocol (IP). The previous example of ABC in the field of wireless transmission is regarded high level. Similar techniques are also applied at considerable lower level. In [van den Bos and Verhoeven, 2001] a method is presented for the design of context-aware analogue radio front-ends. The method merges at run-time homogenous front-end architectures. The methods selects, depending the current channel conditions, an architecture, e.g., super-heterodyne, low IF, direct conversion, etc. In Section 6.2.1 we elaborate on an example of homogeneous merger from the Ubicom project. The case of that section considers graphical objects and their possible representations. Each graphical objects offers an operation space that trades off resource utilisation versus distortion. Larger (compound) objects are constructed from smaller objects. A compound object too offers an operation space and, henceforth can be applied in even larger objects.

Flexible merging of heterogeneous objects (or functions) is an emerging methodology. Typically the component that performs the merger has a choice of services it can employ; the function repertoire. Each of the services offers a particular service and operation space. In order for the merging component to provide a service itself, it requires a minimum amount of these underlying services and resources (epistasis). As an example, a digital television set needs a tuner, a video decoder, and an audio decoder to function properly, although the video decoder is optional when streaming

audio. The actual selection of employed services depends on the objective imposed on the merging component. Design space exploration techniques can be used to evaluated and rank alternative compositions with respect to the context of the merging component. An example of the design of a range radio transceiver front-ends can be found in [Tasic and Serdijn, 2002a]. Tasic and Serdijn propose a design methodology that applies heterogenous merger. The design combines three essential components of a radio transceiver front-end: a low-noise-amplifier, an oscillator, and a mixer (See Figure 2.8 on Page 28). The merging component has a choice of different implementations with different service levels and different operation spaces. Alternatives are valued using so-called K-rail diagrams [Tasic and Serdijn, 2002b]. In Section 6.2.2 we elaborate on an example of heterogenous merger. The case extracts positioning information from a set of otherwise insufficient sensor information.

6.2.1 Homogeneous merger: 3D representation

An example of merging homogeneous components is from the Ubicom project and involves the display of 3D graphical scenes. A scene, typically, is a merger of multiple 3D objects. Displaying a 3D scene requires an accurate projection (mapping) to a 2D display. Obviously, there is a choice here: the representation of a scene and the consequent mapping process can be done more or less accurately. There is a tradeoff in temporal and spatial distortion versus resource utilisation. The tradeoff applies to the scene as a whole, as well as to constituent objects. The concept of abstracting compound objects from its constituents has been pursued in [Pasman and Jansen, 2001], which allowed the authors to develop a context-aware 3D graphics rendering engine.

The computational structure of the rendering system is a distributed one. That is, a scene graph with 3D graphical objects is managed separately from the eventual rendering engine. The rendering engine connects to the display. Figure 6.12 gives some details. A scene graph (scenegraph) is a graph that connects 3D graphical objects. The graphics front-end compiles scenes from the scene graph into an assembled scene graph (assembledsg), which eventually is transferred to the actual rendering engine. The actual mapping (rendering) of the scene to the display depends on the current context. The position (view point) of the user, possible occlusion through real world objects, and environmental conditions such as lighting and contrast are taken into account by the rendering process. Figure 6.12 indicates two exhaustible resources a polygon buffer and a texture buffer. In the end, the filling degree of these buffers determine the power dissipation of the render engine and the refresh rate of these buffers determines the load on the link between front-end and render engine.

The 3D graphics scene graph is an active one. Each graphical object has various methods to generate so-called level-of-detail representations: imposter, meshed imposter, or polygon model [Pasman, 1999]. Each representation is associated with function that describes the relative distortion and the corresponding resource utilisation. As an example, the distortion δ_i of an object o_i depends on the number of polygons N_i of a particular representation. We have

$$\delta_i = \left(\frac{k_i}{N_i}\right)^{p_i} \tag{6.11}$$

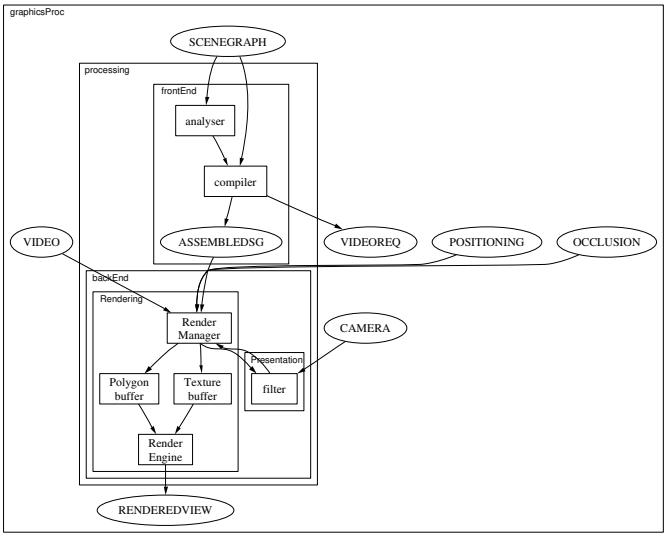


Figure 6.12: Graphics processing; computational view.

The constant k_i and p_i depend on the method of representation. The visual distortion of an object in a scene depends on its location (close range or distant), its size, and its purpose (focus or decoration). The actual visual distortion is a scaled version of the relative distortion; scaled with the angle subtended (covered) by the object. Let r be the radius of the enveloping sphere of the object and d the distance to the object then the visual distortion δ_i^* is given as

$$\delta_i^* = \arctan\left(\frac{r}{d}\right)\delta_i \tag{6.12}$$

The domain of the relative distortion function is bounded. There is a minimum number of required polygons to represent an object with a high distortion measure that is just acceptable. At the other end of the domain the distortion will reach a minimum possible value, irrespective of the number of additional polygons. Figure 6.13 has two objects with each a specific representation mode. Their operation spaces are piecewise linear on a log-log scale.

From a perceptive point of view, individual objects in the visible scene preferably have comparable, relative, distortion measures. Hence, a composition of graphical objects is again a graphical object with multiple representation methods and corresponding distortion and resource utilisation values. The underlying distortion model assures a balanced relative distortion of the embedded objects. The merger of two objects with a given operation space is a simple addition of their operation spaces, which fits the piecewise linear model of Equation (6.11). The resulting compound object has, in log space, a piecewise linear operation space. A mathematical treatment of a similar merger process has been described in Section 4.4.

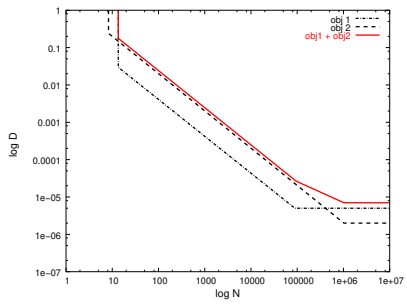


Figure 6.13: Merger of two objects.

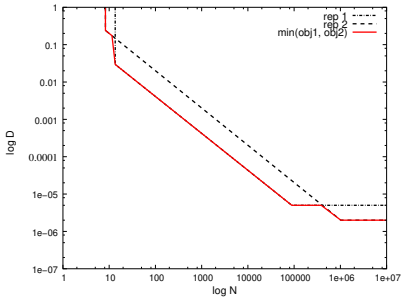


Figure 6.14: Optimal representation of an object.

The required throughput of the link is an important resource utilisation parameter. The workload on this resource, obviously, depends on the complexity of the representation of an object as well as the so-called life-time of the representation. The life-time of the representation of an object that resides in the render engine is determined by the dynamics of the scene. Both the application and the user may initiate actions that affect the relative importance of an object. Obviously, when complex representations are applied for focused objects as well as for decorative objects, this extends the life-time of the representation. Although complex representations tend to lower the constraints on throughput demand, complex representations also put a significant workload on storage and render capacity of the render engine.

The scene graph offers an operation space for objects and compound objects. The offer is based on a guessed position in order to generate a visual distortion measure. Optimisation of a set of possible representations is straightforward. For each targeted distortion value the minimum value is taken. See Figure 6.14 for an example.

The operation space of the scene graph has a third dimension. Next to the number of polygons and the visual distortion there is a habitat: an area in which an representation of an object can guarantee the distortion measure. In this particular case a conic range is used; a cone with spheric intersections that bound the area. Because of the shape of this area compound objects always consist of objects that are in the vicinity of each other. Given the operation space, the render engine can trade off the load on the link (throughput), the load on the render engine (number of polygons), and the load on the texture memory. Typical real-time rendering engines, which must keep up with head movements of the user, are limited to render only a few thousand polygons per frame.

6.2.2 Heterogeneous merger: sensor fusion

Sensor (or data) fusion is a typical example of a heterogeneous merger. Data fusion is applied in the Ubicom project for the recovery of the position and orientation of the user's head. This so-called "pose recovery", analyses and combines incomplete, and possibly inaccurate, data from multiple sensing devices. The context of the positioning component is a varying one. The application has varying objectives with respect to the

accuracy of the derived position. Also the conditions vary; availability of resources, time, etc.

We will illustrate a heterogeneous merger, by means of a simulation model. Assume we have available a set of sensors and corresponding functionality to analyse the raw sensor data. Each sensor has a fixed sampling period and a fixed mapping of its analysis functionality (computational entity) to a resource (engineering entity). Each sensor observation henceforth is a partial, perturbed, and delayed one. In that, 1) only part of the position and location information is measured, 2) perturbations have a known probability distribution, 3) observations have a lag, and 4) observations and their analysis have associated a resource utilisation vector. Note that we presume that the perturbation of the sensor data is free of *drift*. This is an oversimplification of real-life systems. A system that suffers from drift has to estimate second order statistics of perturbations in addition to the first order statistics included in the model presented here.

Our case study considers 2D position information ($\hat{x}$, $\hat{y}$) and heading ($\hat{\alpha}$) information. The sensor fusion process combines partial and incorrect data from individual sensors so as to arrive at an estimate of the current position. The estimate of the position is based on a simple model combining measurements of $\langle x, y, \alpha \rangle$ and their first derivatives, $\langle v_x, v_y, \omega \rangle$. Let t be the time evolution since the last position update, then an estimate of the current position – the position model – is given as:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\alpha} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \alpha \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{6.13}$$

The fusion process applies a linear Kalman filter [Kailath et al., 2000]. The Kalman filter tracks the status and the time evaluation of the model parameters. Measurements and time updates are incorporated using so-called, Schmidt, forward recursions. Whereas delayed observations (lag) are incorporated using, reversed Schmidt, backward recursions. In order for the Kalman filter to properly track time-evaluations of state parameters, sensors must specify stochastic properties of their observations: the mean and variance of their (systematic) error.

The appeal of implementing a Kalman filter is the ease of combining heterogeneous data: offering updates for a different subset of state parameters, at different (irregular) time intervals, with different accuracy, and different lag. A relevant experiment is to explore the design space while evaluating different combinations of sensors. We take a predefined trajectory and corresponding orientation of a user. We emulate the corresponding measurements of the respective sensors and we emulate their analysis process to transform the measurements to the parameters of the position model. The Kalman filter combines the measurements of each sensor from the set. Causal sampling (omitting future updates) of the position model thus gives an estimated track and estimated orientation for a range of t. Two absolute distortion measures are defined to value the estimated track and orientation with respect to the actual track and orientation. D_{pos} is the Euclidean distance between the estimated and actual track and D_{head} is the angular distance between the estimated orientation and the actual orientation.

The set of individual sensors and their characteristics are summarised in Table 6.9. The table presents for each sensor the state parameters for which updates are sup-

Table 6.9: Sensor characteristics

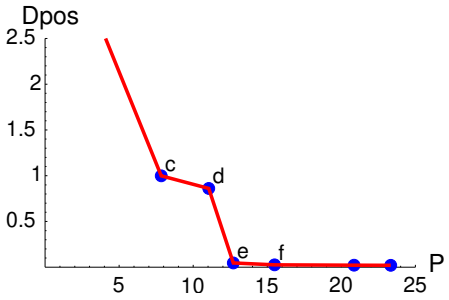
sensor	Id	state	T	T_{lag}	P	(μ, σ)
local image based pose-recovery	ImgI	(x, y, α)	32	8	3.21	(0, 0.0015)
remote image based pose-recovery	ImgO	(x, y, α)	64	16	0.43	(0, 0.00015)
Global Positioning System	GPS	(x, y)	16	4	3.66	(0, 3.75)
Differential GPS	DGPS	(x, y)	12	3	6.13	(0, 0.8)
Accelerometer	Acc	(v_x, v_y)	4	1	3.43	(0, 0.015)
Gyroscope	Gyro	(ω)	4	1	8.85	(0, 0.015)
magneto-meter	Mag	(α)	8	2	1.71	(0, 0.12)

plied, the sampling interval (T) at which updates become available, and the delay (T_{lag}) between the actual measurement and the availability of the measurement for the Kalman filter. Measurements are perturbed with Gaussian noise with a mean μ and variance σ. The power parameter, P, specifies the required power to implement the analysis. Take for instance the image based pose-recovery components, ImgI and ImgO. The component captures and analyses an image from a head-mounted camera. Analysis of the image involves matching it with a database of pre-recorded image characteristics. There are two versions of the component in the set of sensors: ImgI that implements the analysis process locally and ImgO that implements the analysis process remotely. Obviously local analysis can be implemented faster and more frequently, yet dissipates more power and is less accurate than remote implementation, see Table 6.9 for details.

The design space is being explored through a series of experiments. Each experiment combines observations from a subset of the available sensors. The overall, system, distortion is measured as the variance of D_{pos}. In Figure 6.15 we plot the non-inferior operation points with a tradeoff between distortion and power dissipation. Interestingly, image based pose recovery with remote analysis (ImgO) offers the best possible position estimate, yet in the context of the experiment, its superior behaviour over local analysis (ImgI) is negligible; c.f. sets `e` and `f` of Figure 6.15.

The pose-recovery system offers an operation space that allows for a tradeoff distortion versus resource utilisation, power dissipation and link throughput. For the sake of illustration, latency is ignored here. Functionally the pose recovery offers a continuous position model.

The operation space of the pose recovery system can be seamlessly integrated in an ARC framework. Here we applied emulation and simulation to derive the operation space. Ultimately the operation space can be derived from underlying sensor components. Above, we addressed the issue of selecting between an image based position recovery system with local (ImgI) or remote (ImgI) data analysis, both having their pros and cons. Likewise, Table 6.9 suggests we have to choose whether or not to enhance a GPS sensor system (DGPS). In fact, Table 6.9 gives the ARC compliant operation



c: { DGPS, Mag }; d: { ImgI, DGPS, Mag }; e: { ImgO, Acc, Gyro }; f: { ImgI, Acc, Gyro }.

Figure 6.15: Operation space of the position and orientation component.

spaces of possible servers of the pose-recovery system.

6.3 ARC interfaces: Ubicom

The Ubicom system, as previously presented in Section 2.2, uses ARC for coordinating distortion, capacity, and resource utilisation. In this section we present an overview of the ARC interface specifications as found in the Ubicom system. An interface consists of a function repertoire and corresponding operations space that captures the functional and non-functional behaviour of the component. In this section we concentrate on the operations spaces of the components of the communication support system

In Section 2.2 the Ubicom system has been presented to some extent. We took different views on the system which resulted in different diagrams each addressing specific concerns. In this section we take a broad view while concerning the ARC interfaces. The view basically extends the computational view with constructs from the informational view and the engineering view.

The support system of Ubicom (Figure 2.7 on Page 28) comprises components usually found in communication systems. Functionally, the radio front-end (radioFe) takes in a physical channel and outputs, after filtering and conversion, a sampled continuous signal: the continuous channel. The baseband coding component (basebandCodec) transforms the continuous signal into a discrete signal (a bitstream): the raw channel. The channel coding component (channelCodec) exploits the redundancy of information aspect of the raw channel signal so as to improve the reliability of individual bits. The resulting stream is referred as the raw link. Finally, the protocol component (protocols) manages late quality requirements, these efforts result in a set of concurrent (possibly unreliable) links available to the rest of the system. See Figure 6.16 for an abstract computational view on the support system.

The informational view on the support (communication) system identifies the interfacing objects between cooperating components as tuples. Each tuple includes a sending and a receiving object. The parameters of each tuple corresponds with the ARC interface parameters. Tables 6.10 through 6.15 specify the respective ARC inter-

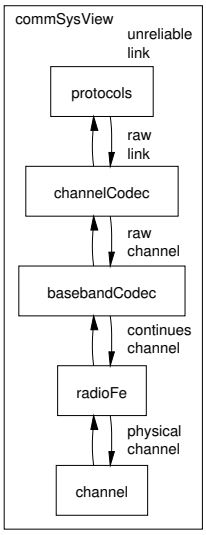


Figure 6.16: Ubicom support system, computational view.

	Description
$P_{\max}$	maximum allowed transmit power
d	sender to receiver distance
τ_{rms}	channels multipath characterisation
N_o	background noise level

Table 6.10: ARC-interface of the (physical) channel; ⟨RADIOSND, RADIORCV⟩.

	Description
p	outage probability
SNR	signal to noise ratio
P	power dissipation
BW	bandwidth

Table 6.11: ARC-interface of the continuous channel; ⟨BASEBANDSND, BASEBANDRCV⟩.

faces. The tuples refer to the informational view constructs of Figure 2.7 on Page 28. Note that the ARC interface of Table 6.15 is not shown in Figure 6.16.

The described ARC interfaces among components and the implicit functionality of the respective components requires for some explanation. Initially the informational partition of the support system followed the traditional demarcation lines between disciplines. In the course of the project's phases options and difficulties became explicit. Initially vaguely posed ARC interfaces became manifest but also evolved. For instance, it is not common practise to functionally separate FEC and ARQ. In case of the Ubicom support system, FEC is implemented by the channel codec whereas ARQ is implemented by the protocols component.

Evolution of ARC interfaces may be caused by newly derived practical insights but also from newly developed theoretical insight. In [van der Schaaf et al., 2001] the ARC

	Description
P	power dissipation
P_{BER}	error probability
T	capacity/throughput
$\mathfrak{L}$	latency

Table 6.12: ARC-interface of the raw channel; ⟨CHANNELSND, CHANNELRCV⟩.

	Description
P	power dissipation
T	throughput
BER	error probability
$\mathfrak{L}$	latency

Table 6.13: ARC-interface of the raw link; ⟨PROTSND, PROTRCV⟩.

	Description
P	power dissipation
CPU	CPU workload
T	capacity/throughput
BER	error probability
$\mathfrak{L}$	latency

Table 6.14: ARC-interface of the (unreliable) link; ⟨TRANSMITREQ, TRANSMIT⟩.

	Description
P	power dissipation
CPU	CPU workload
D	distortion (PSNR)
$\mathfrak{L}$	latency

Table 6.15: ARC-interface of the video link; ⟨VIDEOREQ, VIDEO⟩ .

interface between source coding and the support system ((unreliable) link) is designed from a theoretical perspective. The following practical implementation of this ARC interface (Table 6.14) and that of a source coder (Table 6.15) has been described in Section 6.1.3.

Evolution of ARC interfaces may also yield a shift of functionality. Consider as an example the interface between channel coding and baseband processing. Modern baseband decoding components apply a technique called soft-decoding. Instead, of just offering bits of information to the channel codec, information is augmented with a reliability interval. This way the channel codec is left the opportunity to extract more information from the raw channel than without these second order statistical information. Of course there is a tradeoff, the amount of added statistical information, its accuracy, and the relevance to the channel coder is subject to different objectives. In the Ubicom project, the channel codec considers so-called turbo decoders. Derivation of the operation space of a these coders is notoriously difficult. A new simluation technique based on *importance sampling* is under construction [Bohdanowicz, 2001; Bohdanowicz and Weber, 2002] to value the effect of selecting different modes of operations of the decoder. With the use of this technique, it is possible to derive operations spaces more efficiently.

Seven

Conclusions

In this dissertation we addressed the problem of proper coordination among components in a communicating system. The Ubicom system, a system for mobile communication based on visual augmented reality, provides a real-life example system to which concepts can be verified. Communication systems are notoriously complex because of their scale, the diversity of involved technology domains, their irregular interaction, and last but not least their inherent requirement of being context-aware; a communication system is in a constant state of flux.

In our problem description on Page 4 we identified the prerequisites for the organisation of proper coordination of components. The coordination system should be distributed, non-iterative, and evolutionary. Distributed because the complexity of the underlying system prohibits a practical development of a centralised, omniscient, controller. The non-iterative constraint also originates from practical considerations. For the timely development of solutions the number of iterations have to be bound. The evolutionary constraint stems from the requirement to seamlessly support the incorporation of newly derived results and the adequate reaction to a changing context (varying conditions or changing objectives)

While investigating the problem it becomes apparent that the development of these type of complex systems, involve development parameters that address functionality, flexibility, and resource utilisation. The latter two addressing the non-functional aspects of system development. In the remainder of this chapter we first discuss the results of our work followed with a reflection and a discussion for future improvements.

7.1 Results

In this dissertation we presented a development framework for mastering the complexity of communication systems. The framework recognises three main components for system development:

1. Compositionality; ideally, complex systems are constructed from less complex subsystems. However, compositionality is only guaranteed under determinate

conditions. The composition of indeterminate subsystems yield an indeterminate system, unless the context dependent behaviour of the subsystems can be adequately captured and combined. A general complex system thus involves its functional input/output behaviour as well as its non-functional context dependent behaviour.

2. Communication; multidisciplinary systems require clear communication between subsystems. Objectives and conditions must be conveyed clearly in order to exchange options and limitations among subsystems.

3. Coordination; sharing of resources requires coordination. Multiple ways exist to organise this coordination. We have choosen an heterarchic approach, so as to circumvent developing an knowing it all central entity. An important argument in favour of a distributed and non-hierarchical organisation of the coordination is to design for flexibility.

We adopted a systemic ontology, which realises a set of related views on the system (Chapter 2); each view emphasising specific concerns. The ontology predicates system properties in relation to the immediate environment (context) of the system. The scope and detail of a view varies with its purpose. Some views, like the enterprise view, comprise the entire system whereas other views only serve the purpose of enabling clear local communications. Ambiguity among views is not an issue since each view articulates the shared understanding of the involved stakeholders.

In Postulate 1.1 on Page 6 we argued against the development of an all comprising meta framework. We demonstrated that irrespective of a more practical oriented issues it is very hard to develop and maintain a consistent metaframework for systems that are in a constant state of flux.

For a clear conveyance of information between subsystems, the subsystem developers must share a context. Fruitful collaboration among developers of neighbouring subsystems thus requires context-awareness among these developers and their subsystems. We have formalised this concept in the ARC framework (Chapter 3) that implements abstraction, adaptation, and cooperation. Abstraction is the usual way of communicating behavioural information between subsystems. An abstraction on communicating systems that includes non-functional aspects is usually a tuple of distortion, capacity, and resource utilisation. Adaption is an inbound awareness of a variable context, adding an outbound awareness requires cooperation. The outbound awareness is due to the fact that subsystems share the responsibility for the overall system integrity. Since the system is in a constant state of flux, subsystems experience an highly-variable immediate environment. To capture the variability, information from the immediate environment is a great asset. Since two neighbouring subsystems are in each other's milieu, cooperation is beneficial from both perspectives. An interesting by-product of the ARC framework is a conceptual framework for negotiated QoS, which allowed us to do a systematic comparison of existing frameworks for QoS.

The ARC framework implements the requirement of Postulate 1.2 on Page 8 that a practical coordination system should implement a non-iterative and distributed control structure. Distributed control or coordination is possible because system components

implement a so-called oracle, which captures the context-awareness of the component and the necessary domain expertise to make this effective.

Part of the functionality of the oracle solves in essence a multiobjective optimisation problem (Chapter 4). The optimisation problem is structured in the sense that one encounters a typical set of operation space parameters ($\langle \boldsymbol{\delta}, \boldsymbol{\tau}, \boldsymbol{\rho} \rangle$) and decision space boundaries (Decision space, Consistency, and Compatibility). There is however no general solution to the emerging optimisation problem, we referred to literature for a selection of solutions to known problems.

With ARC we created a bi-directional context awareness, which implies the isolation of the context-dependent behaviour of a subsystem. Because of this isolation the system becomes compositional. The CAPN model of computation is a dedicated implementation of the ARC concepts, namely for stream-based applications with occasional event handling (Chapter 5). For a specific subtype of these applications CAPN can derive the conditions on the context of a subsystem for its proper operation. Or vice versa, analyse the system behaviour for a given (parameterised) context. In Postulate 1.3 on Page 10 we required non-functional information to be available on the interfaces of a subsystem and oraclisation of the indeterminate part of a subsystem. The ARC framework implements both requirements, whereas the CAPN model of computation makes the oraclisation explicit and formalises compositionality.

The concepts of our development framework have been illustrated with various case studies where we assessed their agility and efficacy.

7.2 Reflection

Coordination of shared resources in complex systems is an active area of research and development. Industry is taking up and gradually introduces Quality of Service (QoS) is their systems [Cisco Systems, 2003; Remondo and Niemegeers, 2003]. They are supported by standardisation bodies who are committed to make QoS an integral part of their work [Blake et al., 1998; Al-Karaki and Chang, 2004; Adis, 2003]. In the mean time, research considers the next generation of complex systems, such ad-hoc and peer-to-peer networks that combine concurrency, QoS, wirelesss communication, and context awareness [see, e.g, Pouwelse et al., 2004; Liu and Issarny, 2004].

Although current industrial implementations of QoS are contract based the contracts are usually not negotiable; contracts are only established at service set up time. The server is committed to deliver the agreed level of quality, whereas the client is committed to acknowledge the contract. The established level of quality of service therefore is a statically defined best-effort service. In a commercial setting policing and monitoring appears to be an essential requirement. In [Cisco Systems, 2003] an implementation based on differentiated services (diffServ) [Blake et al., 1998] is described for Voice Over IP and Video on Demand services. Both applications use the Internet but rely on minimum guarantees for proper operation.

The hesitation to implement QoS on a great scale has two main sources. First, QoS inherently introduces overhead which can be prohibitively expensive [Adis, 2003] and second QoS is inherently difficult because client and servers need to come to a mutual understanding of their requirements and offers [Duran-Limon and Blair, 2004].

New schemes for negotiated QoS and experiments with these and existing schemes are published regularly, e.g., [Jukic et al., 2004; Al-Karaki and Chang, 2004]. Our ARC framework has recently successfully been applied in a video streaming application over a wireless link [Taal et al., 2003].

In Chapter 2 we emphasised the need for multiple views on a system, the upcoming IEEE standard for architectural descriptions [IEEE-1471, 2000] is by now fully accepted by the software architecture research community [Clements et al., 2003; Bosch, 2004]. The selection of views and their implementation remains however a concern. It turns out that generic views that have a wide scope are without exception ambiguous, therefore industry is rather reluctant in accepting this technology [Graaf et al., 2003].

Our CAPN model of computation seems a promising approach for system construction. Advanced methods for developing dependable systems currently rely on the determinacy of the system components [Stefanov et al., 2004; Lee, 2001]. The CAPN is candidate model of computation to introduce a controlled level of indeterminacy, which increases the applicability of these development methods in practice.

7.3 Future improvements

This dissertation takes the perspective of a system architect and consequently involves a wide range of domains of expertise. There is room for improvement on the individual domains as well as on the field of collaboration among domains.

We use views to communicate concerns among stakeholders. The concept of a view is materialised in Bunge's systematic position. An improvement on the current situation is to structure the views formally. This would yield an opportunity for doing systematic analysis of views. The Bunge-Wand-Weber ontological model, as presented in Section 2.1.3, is potentially a good starting point.

The design-time and run-time experiments we conducted with the ARC framework were manually casted (Chapter 6). The interfaces, the optimisations, and the evaluation routines were all made specific for a subsystem. For a dedicated application area it should be possible though to construct a technology that implements the ARC interface routines and offers flexible optimisation and coordination routines. For a dedicated application area it is possible to implement the three-sweep ARC interface protocol: request, offer, select; only the number and interpretation of interface parameters vary. Also for a dedicated application area one can devise a library of ARC coordination and optimisation routines. Many components execute similar optimisation routines. For instance, when a design space is sufficiently small, an exhaustive exploration becomes feasible; only the evaluation function is component specific. A centralised implementation of this type of coordination (perfd) has been considered in [Pouwelse, 2003].

Our CAPN model of computation ultimately captures the context dependent behaviour of a system in a control stream. This is a primitive way of modelling context-dependent behaviour. Research is required to develop more generally applicable methods for modelling context-dependent behaviour.

Bibliography

[van der Aalst, 1999] W. van der Aalst. "Formalization and verification of event-driven process chains." *Information and Software Technology*, vol. 41:pp. 639–50, 1999.

[Abdelzaher et al., 2002] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. "Performance guarantees for Web server end-systems: A control-theoretical approach." *IEEE Transactions on Parallel and Distributed Systems*, vol. 13(1):pp. 80–96, 2002.

[Abraham and Rau, 2000] S. G. Abraham and B. R. Rau. "Fast automated design space exploration in PICO." In *Int. Conf. for Compilers, Architecture, and Synthesis for Embedded systems (CASES 2000)*. San Jose (CA), 2000.

[Adis, 2003] W. Adis. "Quality of service middleware." *Industrial Management & Data Systems*, vol. 103(1):pp. 47–51, 2003.

[Aditya and Rau, 2000] S. Aditya and B. R. Rau. "Automatic architecture synthesis and compiler retargeting for VLIW and EPIC processors." Tech. Rep. HPL-1999-93, HP Laboratories Palo Alto, Jan. 2000.

[Agha, 2002] G. A. Agha. "Introduction." *Communications of the ACM*, vol. 45(6):pp. 30–32, 2002. Special issue on Adaptive middleware.

[Al-Karaki and Chang, 2004] J. N. Al-Karaki and J. M. Chang. "Quality of service support in IEEE 802.11 wireless ad hoc networks." *Ad Hoc Networks*, vol. 2:pp. 265–281, 2004.

[Alexandrov and Lewis, 2000a] N. Alexandrov and R. Lewis. "Algorithmic perspectives on problem formulations in MDO." In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 8th, Long Beach, CA*. 2000a.

[Alexandrov and Lewis, 2000b] N. M. Alexandrov and R. M. Lewis. "Analytical and computational properties of distributed approaches to MDO." In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 8th, Long Beach, CA*. 2000b.

[Andrade and Fiadeiro, 2001] L. F. Andrade and J. L. Fiadeiro. "Coordination: The evolutionary dimension." In *Proceedings of the Conference on Technology of Object Oriented Languages and Systems (TOOLS 38)*, pp. 136–147. 2001.

[AOSD-web, 2001–2002] AOSD-web. "Aspect oriented software development." `http://aosd.net/`, 2001–2002.

[Apt and Plotkin, 1986] K. R. Apt and G. D. Plotkin. "Countable nondeterminism and random assignment." *Journal of the ACM (JACM)*, vol. 33(4):pp. 724–767, 1986.

[Aurrecoechea et al., 1999] C. Aurrecoechea, A. T. Campbell, and L. Hauw. "A survey of QoS architectures." In *7th Int. Workshop on Quality of Service (IWQoS'99)*. 1999.

[Bakker et al., 2001] J.-D. Bakker, K. Langendoen, and H. Sips. "LART: Flexible, low-power building blocks for wearable computers." In *Int. Workshop on Smart Appliances and Wearable Computing (IWSAWC)*. Scottsdale, AZ, 2001.

[Balling and Rawlings, 2000] R. Balling and M. Rawlings. "Collaborative optimization with disciplinary conceptual design." *Structural-and-multidisciplinary-optimization*, vol. 20(3):pp. 231–241, Nov. 2000.

[Bertsekas, 1999] D. P. Bertsekas. *Nonlinear programming*. Belmont : Athena Scientific, 1999.

[Boijmans Van Beuningen, 1849,1958] Boijmans Van Beuningen. "Museum Boijmans Van Beuningen." 1849,1958. Rotterdam, The Netherlands.

[Bézivin and Gerbé, 2001] J. Bézivin and O. Gerbé. "Towards a precise definition of the OMG/MDA framework." In *Proceedings 16th Annual International Conference on Automated Software Engineering*, pp. 273–80. ASE 2001, 2001.

[Bhattacharya and Bhattacharyya, 2001] B. Bhattacharya and S. Bhattacharyya. "Parameterized dataflow modeling for DSP systems." *IEEE Transactions on Signal Processing*, vol. 49(10):pp. 2408–21, 2001.

[Bhattacharyya et al., 1999] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee. "Synthesis of embedded software from synchronous dataflow specifications." *Journal of VLSI Signal Processing Systems*, vol. 21(2):pp. 151–166, Jun. 1999.

[Bhatti and Knight, 1999] S. Bhatti and G. Knight. "Enabling QoS adaptation decisions for Internet applications." *Journal of Computer Networks*, vol. 31(7):pp. 669–692, Mar. 1999.

[Bianchi et al., 1998] G. Bianchi, A. Campbell, and R.-F. Liao. "On utility-fair adaptive service in wireless network." In *6th Int. Workshop on Quality of Service (IWQoS'98)*, pp. 256 – 267. 1998.

[Bilsen et al., 1996] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete. "Cyclo-static dataflow." *IEEE Transactions on Signal Processing*, vol. 44(2):pp. 397–408, Feb. 1996.

[Blake et al., 1998] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An architecture for differentiated services." RFC 2475, IETF, Dec. 1998.

[Bohdanowicz, 2000] A. Bohdanowicz. "Wideband indoor and outdoor radio channel measurements at 17 GHz." Tech. Rep. UbiCom-TechnicalReport/2000/2, Delft University of Technology, 2000.

[Bohdanowicz, 2001] A. Bohdanowicz. "On efficient BER evaluation of digital communication systems via importance sampling." In *Proceedings 8th Symposium on Communications and Vehicular Technology*, pp. 61–67. Delft, The Netherlands, 2001.

[Bohdanowicz and Weber, 2002] A. Bohdanowicz and J. Weber. "Conceptual framework of integrated variance reduction techniques for performance evaluation of communication systems." In *Proc. of the IASTED International Conference on Communications and Computer Networks*, pp. 335–341. CCN 2002, Cambridge, MA, USA, 2002.

[van den Bos, 1999] C. van den Bos. "About non-linear distortion of OFDM signals." In *Proceedings of ProRISC*. 1999.

[van den Bos and Verhoeven, 2001] C. van den Bos and C. Verhoeven. "Reconfigurable radio receiver front ends for mobile communications." *Tijdschrift van het Nederlands Elektronica en Radiogenootschap*, vol. 66:pp. 99–104, 2001.

[Bosch, 2004] J. Bosch. "Software architecture: The next step." In F. Oquendo, B. Warboys, and R. Morisson, (eds.) *Proceedings of the first European workshop on Software Architecture (EWSA)*, LCNS 3047, pp. 194–199. 2004.

[Braithwaite, 1962] R. Braithwaite. "Introduction to Gödel's theorem." `http://www.ddc.net/ygg/etext/godel/godel2.htm`, 1962.

[Braun and Kroo, 1997] R. D. Braun and I. M. Kroo. "Development and application of the collaborative optimization architecture in a multidisciplinary design environment." In *Multidisciplinary design optimization State of the art; Proceedings of the ICASE/NASA Langley Workshop, Hampton, VA*, pp. 98–116. 1997.

[Brock and Ackermann, 1981] J. D. Brock and W. B. Ackermann. "Scenarios: A model of nondeterminate computation." In J. Diaz and I. Ramos, (eds.) *Formalization of programming*

concepts, Lecture Notes in Computer Science, pp. 225–259. Springer, 1981, 107 edn.

[de Bruin and Nienhuys-Cheng, 1998] A. de Bruin and S. H. Nienhuys-Cheng. "Linear dynamic Kahn networks are deterministic." In *Theoretical Computer Science*, pp. 3–32. 1998.

[Buck, 1993] J. T. Buck. *Scheduling dynamic dataflow graphs with bounded memory using the token flow model*. Ph.D. thesis, University of California at Berkeley, 1993.

[Bunge, 1977] M. Bunge. *Treatise on basic philosophy. Vol. 3. Ontology I: The furniture of the world*. Reidel, Dordrecht, 1977.

[Bunge, 1979] M. Bunge. *Treatise on basic philosophy. Vol. 4. Ontology II: A world of systems*. Reidel, Dordrecht, 1979.

[Cassandras and Lafortune, 1999] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Boston Kluwer Academic, 1999.

[Chatterjee et al., 1997] S. Chatterjee, J. Sydir, B. Sabata, and T. Laurance. "Modeling applications for adaptive QoS-based resource management." In *High Assurance System Engineering workshop (HASE'97)*. 1997.

[Chen et al., 2001] H.-S. A. Chen, L. Qiao, and K. Nahrstedt. "Adaptive versus reservation-based synchronization protocols-analysis and comparison." *Multimedia Tools and Applications*, vol. 14(3):pp. 219–57, 2001.

[Cisco Systems, 2003] Cisco Systems, (ed.). *Internetworking Technology Handbook*, chap. Quality of Service Networking. Cisco systems, 2003.

[Clements et al., 2003] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting software architectures; views and beyond*. Addison-Wesley, 2003.

[Coello Coello, 1999] C. A. Coello Coello. "A comprehensive survey of evolutionary-based multiobjective optimization techniques." *Knowledge and Information Systems*, vol. 1(3):pp. 269–308, 1999.

[Coello Coello and Christiansen, 1999] C. A. Coello Coello and A. D. Christiansen. "MOSES: A multiobjective optimization tool for engineering design." *Engineering Optimization*, vol. 31(3):pp. 337–368, 1999.

[Crochiere and Rabiner, 1981] R. E. Crochiere and L. R. Rabiner. "Interpolation and decimation of digital signals-a tutorial review." *Proceedings of the IEEE*, vol. 69:pp. 300–31, 1981.

[Crumley, 1995] C. L. Crumley. "Heterarchy and the analysis of complex societies." In R. M. Ehrenreich, C. L. Crumley, and J. E. Levy, (eds.) *Heterarchy and the Analysis of Complex Societies*, pp. 1–5. American Anthropological Association, 1995.

[Czarnecki and Eisenecker, 2000] K. Czarnecki and U. Eisenecker. *Generative programming: methods, tools, and applications*. Addison-Wesley, 2000.

[David and Alla, 2001] R. David and H. Alla. "On hybrid petri nets." *Discrete Event Dynamic Systems*, vol. 11(1–2):pp. 9–40, 2001.

[Deprettere et al., 2002] E. F. Deprettere, E. Rijpkema, and B. Kienhuis. "Translating imperative affine nested loop programs into process networks." *Lecture Notes in Computer Science*, vol. 2268:pp. 89–113, 2002.

[van Dijk et al., 2000a] H. van Dijk, K. Langendoen, and H. Sips. "Application of ARC in system design." In *2nd Int. Symposium on Mobile Multimedia Systems & Applications (MMSA'2000)*, pp. 118–125. Delft, The Netherlands, 2000a.

[van Dijk et al., 2000b] H. van Dijk, K. Langendoen, and H. Sips. "ARC: a bottom-up approach to negotiated QoS." In *3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, pp. 128–137. Monterey, CA, 2000b.

[van Dijk and van Reeuwijk, 1999] H. W. van Dijk and K. van Reeuwijk. "The first Ubicom prototype (blue book)." Technical Report 1999/1, Ubiquitous Communications, Feb. 1999.

[van Dijk et al., 2003] H. W. van Dijk, H. J. Sips, and E. F. Deprettere. "Context aware process networks." In *Proc. of Int. Conf. on Application-specific Systems, Architectures and Processors*,

pp. 6–16. ASAP 2003, 2003.

[Dijkhoff et al., 2000] O. Dijkhoff, V. D. A. Petrignani and, and J. Braat. "Dual Purkinje imaging gaze tracker using a smart vision sensor." In *Proceeding 2nd International symposium on mobile multimedia systems and applications*, pp. 48–55. MMSA 2000, 2000.

[Dijkstra, 1976] E. W. Dijkstra. *A discipline of programming*. Prentice-Hall, Englewood Cliffs, N.J., 1976.

[Dilts et al., 1991] D. Dilts, N. Boyd, and H. Whorms. "The evolution of control architectures for automated manufacturing." *Journal of Manufacturing Systems*, vol. 10(1):pp. 79–93, 1991.

[Dorf, 1989] R. C. Dorf. *Modern Control Systems*. Addison-Wesley, 1989, fifth edn.

[Dou, 1652] G. Dou. "De kwakzalver (The Quack)." oil on panel, 1652. Owned by Boijmans Van Beuningen [1849,1958].

[Duran-Limon and Blair, 2004] H. A. Duran-Limon and G. S. Blair. "QoS management specifciation support for multimedia middleware." *The journal of Systems and Software*, vol. 72:pp. 1–23, 2004.

[Edwards et al., 1997] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli. "Design of embedded systems: Formal models, validation, and synthesis." *Proceedings of the IEEE*, vol. 85(3):pp. 366–390, Mar. 1997.

[Elrad et al., 2001] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr, and H. Ossher. "Discussing aspects of AOP." *Communications of the ACM*, vol. 44(10):pp. 33–38, 2001. Special issue on Aspect Oriented Programming.

[Foundation, 1984] T. F. S. Foundation. "Gnu's is not Unix." `http://www.fsf.org`, 1984.

[Fowler, 2003] M. Fowler. "Design – who needs an architect?" *Software, IEEE*, vol. 20(5):pp. 11–13, 2003.

[Frodigh et al., 2001] M. Frodigh, S. Parkvall, C. Roobol, P. Johansson, and P. Larsson. "Future-generation wireless networks." *IEEE Personal Communications*, vol. 8(5):pp. 10–17, 2001.

[Frølund and Koistinen, 1998] S. Frølund and J. Koistinen. "Quality-of-service specification in distributed object systems." *Distributed Systems Engineering Journal*, vol. 5(4):pp. 179–202, 1998.

[van Gemund, 1996] A. van Gemund. *Performance Modeling of Parallel Systems*. Ph.D. thesis, Delft University of Technology, Apr. 1996.

[Girod and Farber, 1999] B. Girod and N. Farber. "Wireless video." In M.-T. Sun and A. Reibman, (eds.) *Compressed Video Over Networks*, chap. 12. Marcel Dekker, 1999.

[Goel et al., 1999] A. Goel, D. Steere, C. Pu, and J. Walpole. "Adaptive resource management via modular feedback control." Tech. Rep. CSE-99-03, Oregon Graduate Institute of Science and Technology, Jan. 1999.

[Goldie and Pinch, 1991] C. M. Goldie and R. G. Pinch. *Communication Theory*. No. 20 in London Mathematical Society Student Texts. Cambridge University Press, 1991.

[Gould, 1977] S. Gould. *Ever since Darwin*. Norton, New York, 1977.

[Graaf et al., 2003] B. Graaf, M. Lormans, and H. Toetenel. "Embedded software engineering: state of the practice." *IEEE Software Magazine*, vol. 20(6):pp. 61–69, November–December 2003.

[Green and Rosemann, 2000] P. Green and M. Rosemann. "Integrated process modeling: An ontological evaluation." *Information Systems*, vol. 25(2):pp. 73–87, Apr. 2000.

[Gruber, 1993] T. R. Gruber. "A translation approach to portable ontology specifications." *Knowledge Acquisition*, vol. 5(2):pp. 199–220, 1993.

[Haimes et al., 1989] Y. Y. Haimes, K. Tarvainen, T. Shima, and J. Thadathil. *Hierarchical Multiobjective Analysis of Large-Scale systems*. New York: Hemisphere publishing corporation, 1989.

[Hansen et al., 2001] J. Hansen, J. Lehoczky, and R. Rajkumar. "Optimization of quality of service in dynamic systems." In *Proceedings 15th International Parallel and Distributed Processing Symposium*, pp. 1001–1008. 2001.

[Healey, 1999] R. Healey. "Holism and nonseparability in physics." In E. N. Zalta, (ed.) *The Stanford Encyclopedia of Philosophy*. 1999, fall 1999 edition edn.

[Heusdens et al., 2001] R. Heusdens, R. Vafin, and W. B. Kleijn. "Sinusoidal modeling of audio and speech using psychoacoustic-adaptive matching pursuits." In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3281–3284. (ICASSP'01), 2001.

[Hoare, 1985] C. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985.

[Hofstadter, 1985] D. Hofstadter. *Gödel Escher en Bach: Een eeuwige gouden band*. Amsterdam: Contact, 1985. Dutch translation of Godel, Escher, Bach: an eternal golden braid, 1979.

[Holland, 1992] J. H. Holland. *Adaptation in natural and artificial systems: An introductionary analysis with applications to biology, control and artificial intelligence*. MIT Press/Bradford Books edition, 1992.

[Holloway et al., 1997] L. E. Holloway, B. H. Krogh, and A. Giua. "A survey of Petri Net methods for controlled discrete event systems." *Discrete Event Dynamic Systems*, vol. 7(2):pp. 151–190, 1997.

[Holzmann, 1997] G. J. Holzmann. "The model checker SPIN." *Software Engineering*, vol. 23(5):pp. 279–295, 1997.

[IEEE-1471, 2000] IEEE-1471. *IEEE Recommended practice for architectural description of software-intensive systems*. IEEE, 2000. IEEE Std 1471-2000.

[Introna, 2001] L. Introna. "Cooperation, coordination and interpretation in virtual environments: Some thoughts on working-together." *Cognition, Technology & Work*, vol. 3(2):pp. 101–110, 2001.

[Irvine, 2001] A. D. Irvine. "Bertrand Russell." In E. N. Zalta, (ed.) *The Stanford Encyclopedia of Philosophy*. 2001, fall 2001 edition edn.

[Jorge et al., 2000] H. Jorge, C. Antunes, and A. Martins. "A multiple objective decision support model for the selection of remote load control strategies." *IEEE Trans. on power systems*, vol. 15(2):pp. 865–872, May 2000.

[Jukic et al., 2004] B. Jukic, R. Simon, and W. S. Chang. "Congestion based resource sharing in multi-service networks." *Decision support systems*, vol. 37:pp. 397–413, 2004.

[Kahn, 1974] G. Kahn. "The semantics of a simple language for parallel programming." In *Proceedings of the IFIP Congress 74*. North-Holland Publishing Co., 1974.

[Kahn and MacQueens, 1977] G. Kahn and D. B. MacQueens. "Coroutines and networks of parallel processes." In *Proceedings of the IFIP Congress 77*, pp. 993–998. 1977.

[Kailath et al., 2000] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, 2000.

[Karaivanova et al., 1995] J. Karaivanova, P. Korhonen, S. Narula, J. Wallenius, and V. Vassilev. "A reference direction approach to multiple objective integer linear programming." *European Journal of Operational Research*, vol. 81(1):pp. 176–187, 1995.

[Kiczales et al., 2001] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold. "Getting started with ASPECTJ." *Communications of the ACM*, vol. 44(10):pp. 59–65, Oct. 2001.

[Kiczales et al., 1997] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. "Aspect-oriented programming." In M. Akşit and S. Matsuoka, (eds.) *ECOOP '97 — Object-Oriented Programming 11th European Conference, Jyväskylä, Finland*, vol. 1241, pp. 220–242. Springer-Verlag, New York, NY, 1997.

[Kienhuis, 1999] A. Kienhuis. *Design Space Exploration of Stream-based Dataflow Architecture Methods and Tools*. Ph.D. thesis, Delft University of Technology, Jan. 1999. ISBN 90-5326-029-3.

[Kim et al., 2000] H. M. Kim, N. F. Michelena, T. Jiang, and P. Y. Papalambros. "Target cascading in optimal design." In *Proceedings of Int. Design Engineering technical conference (DETC*

2000), DETC2000/DAC-14265. 2000.

[de Kock et al., 2000] A. de Kock, G. Essink, W. Smits, P. van der Wolf, J.-Y. Brunel, W. Kruijtzer, P. Lieverse, and K. Vissers. "Yapi: Application modeling for signal processing systems." In *Proc. 37th Design Automation Conference (DAC'00)*, pp. 402–405. 2000.

[Kodiyalam and Sobieszczanski, 2000] S. Kodiyalam and S. J. Sobieszczanski. "Bilevel integrated system synthesis with response surfaces." *AIAA*, vol. 38(8):pp. 1479–1485, Aug. 2000.

[Koliver et al., 2002] C. Koliver, K. Nahrstedt, J.-M. Farines, J. d. S. Fraga, and S. A. Sandri. "Specification, mapping and control for QoS adaptation." *Real time systems*, vol. 23(1–2):pp. 143–174, 2002.

[Kon et al., 2002] F. Kon, F. Costa, G. Blair, and R. H. Campbell. "The case for reflective middleware." *Communications of the ACM*, vol. 45(6):pp. 33–38, 2002.

[Kroo and Manning, 2000] I. Kroo and V. Manning. "Collaborative optimization - status and directions." In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 8th, Long Beach, CA*. 2000.

[Kumar et al., 2002] S. Kumar, F. Zhao, and D. Shepherd. "Collaborative signal and information processing in microsensor networks." *IEEE Signal Processing Magazine*, vol. 19(2):pp. 13–14, 2002.

[Kung, 1988] S. Kung. *VLSI Array Processors*. Prentice Hall, 1988.

[Lagendijk, 2000a] R. Lagendijk. "The TU-Delft research program Ubiquitous Communications." In *21st Symp. on Information Theory in the Benelux*. Wassenaar, The Netherlands, 2000a.

[Lagendijk, 2000b] R. Lagendijk. "Ubiquitous Communications updated technical annex." Technical Report 2000/1, Ubiquitous Communications, 2000b.

[Lakshman and Yavatkar, 1996] K. Lakshman and R. Yavatkar. "Adaptive resource management for multimedia applications." In *High-Speed networking for multimedia applications*. Kluwer Academic Publishers, 1996.

[Lan and Tewfik, 1998] T.-H. Lan and A. Tewfik. "Power optimized mode selection for H.263 video coding and wireless communications." In *IEEE Conference on Image Processing, (ICIP 98)*. 1998.

[Lee et al., 1999a] C. Lee, J. Lehoczky, R. Rajkumar, and D. P. Siewiorek. "On quality of service optimization with discrete QoS options." In *IEEE Real Time Technology and Applications Symposium*, pp. 276–286. 1999a.

[Lee et al., 1999b] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen. "A scalable solution to the multi-resource QoS problem." In *20th IEEE Real-Time Systems Symposium*, pp. 315–326. Phoenix, AZ, 1999b.

[Lee and Messerschmitt, 1987] E. Lee and D. Messerschmitt. "Synchronous data flow." *Proceedings of the IEEE*, vol. 75(9):pp. 1235–1245, sep 1987.

[Lee, 2001] E. A. Lee. "Overview of the Ptolemy project." Tech. Rep. Technical Memorandum UCB/ERL M01/11, UC Berkeley, Mar. 2001.

[Lee and Messerschmitt, 1994] E. A. Lee and D. G. Messerschmitt. *Digital communication*. Kluwer Academic Publisher, 1994, second edition edn.

[Lee and Parks, 1995] E. A. Lee and T. M. Parks. "Dataflow process networks." *Proceeding of the IEEE*, pp. 773–799, 1995.

[Lee and Sangiovanni-Vincentelli, 1998] E. A. Lee and A. Sangiovanni-Vincentelli. "A framework for comparing models of computation." *IEEE Transactions on Circuit Aided Design*, vol. 17(12), 1998.

[Lee and Sabata, 1999] W. Lee and B. Sabata. "Admission control and QoS negotiations for soft-real time applications." In *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*. 1999.

[Li and Nahrstedt, 1998] B. Li and K. Nahrstedt. "A control theoretical model for quality of service adaptations." In *1998 Sixth International Workshop on Quality of Service (IWQoS'98)*, pp.

145–53. 1998.

[Li et al., 1999] B. Li, D. Xu, and K. Nahrstedt. "Optimal state prediction for feedback-based qos adaptations." In *1999 Seventh International Workshop on Quality of Service. IWQoS'99*, pp. 37–46. 1999.

[Lieberherr et al., 2001] K. Lieberherr, D. Orleans, and J. Ovlinger. "Aspect-oriented programming with adaptive methods." *Communications of the ACM*, vol. 44(10):pp. 39–41, Oct. 2001.

[Lieverse et al., 1999] P. Lieverse, P. van der Wolf, E. Deprettere, and K. Vissers. "A methodology for architecture exploration of heterogeneous signal processing systems." In *Proceedings of SIPS'99*. 1999.

[Lieverse et al., 2001] P. Lieverse, P. van der Wolf, K. Vissers, and E. Deprettere. "A methodology for architecture exploration of heterogeneous signal processing systems." *Journal of VLSI Signal Processing for Signal, Image and Video Technology*, vol. 29(3):pp. 197–207, 2001.

[Liu and Issarny, 2004] J. Liu and V. Issarny. "QoS-aware service location in mobile ad hoc networks." In *Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM)*, pp. 224–235. 2004.

[Livne, 1999] E. Livne. "Special issue: Multidisciplinary design optimization." *Journal of Aircraft*, vol. 36, Jan. 1999.

[Loyall et al., 1998] J. P. Loyall, R. E. Schantz, J. A. Zinky, and D. E. Bakken. "Specifying and measuring quality of service in distributed object systems." In *Proceedings of the 1st International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*. 1998.

[Maier et al., 2001] M. Maier, D. Emery, and R. Hilliard. "Software architecture: introducing IEEE standard 1471." *Computer*, vol. 34(4):pp. 107–109, Apr. 2001.

[Malone and Crowston, 1994] T. W. Malone and K. Crowston. "The interdisciplinary study of coordination." *ACM Computing Surveys (CSUR)*, vol. 26(1):pp. 87–119, 1994.

[Margulis and Fester, 1991] L. Margulis and R. Fester. *Symbiosis as a source of evolutionary innovation: speciation and morphogenesis*. MIT Press, Cambridge, Massachusetts, 1991.

[McCarthy, 1963] J. McCarthy. "A Basis for a Mathematical Theory of Computation." In P. Braffort and D. Hirschberg, (eds.) *Computer Programming and Formal Systems*, pp. 33–70. North-Holland, Amsterdam, 1963.

[McCulloch, 1945] W. McCulloch. "A heterarchy of values determined by the topology of nervous nets." *Bulletin of Mathematical Bioscience*, vol. 7:pp. 89–93, 1945.

[McNamee et al., 2000] D. McNamee, C. Krasic, K. Li, A. Goel, E. Walthinsen, D. Steere, and J. Walpole. "Control challenges in multi-level adaptive video streaming." In *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 3, pp. 2228–2233. 2000.

[M.D. Ercegovac, 1989] T. L. M.D. Ercegovac. "On-line arithmetic for DSP applications." In *Proceedings of the 32nd Midwest Symposium on Circuits and Systems, 1989*, vol. 1, pp. 365–368. 1989.

[Miettinen, 1999] K. Miettinen. *Nonlinear multiobjective optimization*. Boston: Kluwer Academic, 1999.

[Miller, 1965] R. E. Miller. *Switching theory. Vol. II: Sequential circuits and machines*. New York Wiley, 1965. In particular chapter 10: Speed independent switching circuit theory.

[Montanari and Rossi, 1995] U. Montanari and F. Rossi. "Contextual nets." *Acta Informatica*, vol. 32:pp. 545–96, 1995.

[Moore, 1965] G. E. Moore. "Cramming more components onto integrated circuits." *Electronics Magazine*, vol. 38(8):pp. 114–117, Apr. 1965.

[Morse, 1980] J. Morse. "Reducing the size of the nondominated set: pruning by clustering." *Computers and operations research*, vol. 7(1–2):pp. 55–66, 1980.

[Mottahed and Manoochehri, 2000] B. D. Mottahed and S. Manoochehri. "Optimal design of electronic system utilizing an integrated multidisciplinary process." *IEEE Trans. on advanced packaging*, vol. 23(4):pp. 699–707, Nov. 2000.

[Muller, 2001] G. Muller. "The arisal of a system architect." http://www.extra.research.philips.com/natlab/sysarch/MaturityPaper.pdf, Sep. 2001. Gaudi Project.

[Muller, 2004] G. Muller. *CAFCR: A multi-view Method for embedded systems architecting; Balancing Genericity and Specificity*. Ph.D. thesis, Delft University of Technology, 2004.

[Nahrstedt, 1999] K. Nahrstedt. "To overprovision or to share via QoS-aware resource management?" In *Proceedings. The Eighth International Symposium on High Performance Distributed Computing*, pp. 205–212. 1999.

[Nahrstedt et al., 1998] K. Nahrstedt, H. H. Chu, and S. Narayan. "Qos-aware resource management for distributed multimedia applications." *Journal of High Speed Networks*, vol. 7:pp. 229–57, 1998.

[Nahrstedt et al., 2001] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li. "QoS-aware middleware for ubiquitous and heterogeneous environments." *IEEE Communications Magazine*, vol. 39(11):pp. 140–148, Nov. 2001.

[Nair and Keane, 1999] P. B. Nair and A. J. Keane. "Coevolutionary genetic adaptation - a new paradigm for distributed multidisciplinary design optimization." In *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 40th, St. Louis, MO*, pp. 1910–1919. 1999.

[Nair and Keane, 2000] P. B. Nair and A. J. Keane. "A coevolutionary architecture for distributed optimization of complex coupled systems." *AIAA*, 2000.

[Najjar et al., 1999] W. A. Najjar, E. A. Lee, and G. R. Gao. "Advances in the dataflow computational model." *Parallel Computing*, vol. 25:pp. 1907–1929, 1999.

[Nemhauser and Wolsey, 1988] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.

[Noble et al., 1997] B. Noble, M. Satyanarayanan, D. Narayanan, J. T. on, J. Flinn, and K. Walker. "Agile application-aware adaptation for mobility." In *16th ACM Symposium on Operating System Principles*. St. Malo, France, 1997.

[Oosterhoff and de Ridder, 2000] F. Oosterhoff and H. de Ridder. "Transparent visual presentation: image information and ambiguity." In *Proceeding of Presence 2000*. Delft University of Technology, 2000.

[Oosterom et al., 2002] P. Oosterom, J. Stoter, W. Quak, and S. Zlatanova. "The balance between topology and geometry." In *Proceeding of Spatial Data Handling Symposium*. SDH 2002, 2002.

[Opdahl and Henderson-Sellers, 2001] A. L. Opdahl and B. Henderson-Sellers. "Grounding the {OML} metamodel in ontology." *The Journal of Systems and Software*, vol. 57:pp. 119–143, 2001.

[Ossher and Tarr, 2001] H. Ossher and P. Tarr. "Using multidimensional separation of concerns to (re)shape evolving software." *Communications of the ACM*, vol. 44(10):pp. 43–50, Oct. 2001.

[Pamela, 1993–2002] Pamela. "The Pamela project." www.pds.twi.tudelft.nl/pamela/, 1993–2002.

[Panangaden, 1995] P. Panangaden. "The expressive power of indeterminate primitives in asynchronous computation." In *Foundations of Software Technology and Theoretical Computer Science*, pp. 124–150. 1995.

[Panangaden and Shanbhogue, 1988] P. Panangaden and V. Shanbhogue. "McCarthy's amb cannot implement fair merge." In K. Nori, (ed.) *Proceeding of the 8th conference on foundations of software technology and theoretical computer science*, Lecture Notes in Computer Science 338, pp. 348–363. 1988.

[Panangaden and Shanbhogue, 1992] P. Panangaden and V. Shanbhogue. "The expressive power of indeterminate dataflow primitives." *Information and Computation*, vol. 98(1):pp. 99–131, 1992.

[Panangaden and Stark, 1988] P. Panangaden and E. Stark. "Computations, residuals and the power

of indeterminacy,." In T. Lepisto and A. Salomaa, (eds.) *Proceedings of ICALP 1988*, Lecture Notes in Computer Science 317, pp. 439–454. 1988.

[Papadopoulos and Arbab, 1998] G. Papadopoulos and F. Arbab. "Coordination models and languages." Tech. Rep. SEN-R9834, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, Dec. 1998.

[Papalambros and Michelena, 2000] P. Y. Papalambros and N. F. Michelena. "Trends and challenges in system design optimization." In *International Workshop on Multidisciplinary Design Optimization*. 2000.

[Pareto, 1896] V. Pareto. "Cours d'economie politique." Rouge, Lausanne, Switzerland, 1896.

[Park et al., 2002] C. Park, J. Jung, and S. Ha. "Extended synchronous dataflow for efficient DSP system prototyping." *Design Automation for Embedded Systems*, vol. 6:pp. 295–322, 2002.

[Parks, 1995] T. M. Parks. *Bounded Scheduling of Process Networks*. Ph.D. thesis, University of California at Berkeley, 1995.

[Pasman, 1999] W. Pasman. "Comparing 3-d model descriptions for wireless augmented reality." Tech. Rep. 3, Ubiquitous Communications, http://www.ubicom.tudelft.nl, 1999.

[Pasman and Jansen, 2001] W. Pasman and F. Jansen. "Distributed low-latency rendering for mobile augmented reality." In *Proceedings of International Conference of Augmented Reality*. IEEE and ACM (ISAR 2001), 2001.

[Pasman and Jansen, 2002] W. Pasman and F. Jansen. "Comparing object simplification methods for thin client 3D." *IEEE Transactions on Visualization and Computer Graphics*, 2002.

[Patterson and Hennessy, 1990] D. A. Patterson and J. L. Hennessy. *Computer architecture: a quantitative approach*. San Mateo: Kaufmann, 1990.

[Persa and Jonker, 2000] S. Persa and P. Jonker. "Hybrid tracking system for outdoor augmented reality." In *Proceeding 2nd International symposium on mobile multimedia systems and applications*, pp. 41–47. MMSA 2000, 2000.

[Peterson, 1981] J. Peterson. *Petri net theory and the modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

[Pouwelse et al., 1999] J. Pouwelse, K. Langendoen, and H. Sips. "A feasible low-power augmented-reality terminal." In *2nd IEEE and ACM Int. Workshop on Augmented Reality (IWAR'99)*, pp. 55–63. San Francisco, CA, 1999.

[Pouwelse et al., 2000] J. Pouwelse, K. Langendoen, and H. Sips. "Dynamic voltage scaling on a low-power microprocessor." In *2nd Int. Symposium on Mobile Multimedia Systems & Applications (MMSA'2000)*, pp. 157–164. Delft, The Netherlands, 2000.

[Pouwelse et al., 2001] J. Pouwelse, K. Langendoen, and H. Sips. "Dynamic voltage scaling on a low-power microprocessor." In *7th ACM Int. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 251–259. Rome, Italy, 2001.

[Pouwelse, 2003] J. A. Pouwelse. *Power Management for Portable Devices*. Ph.D. thesis, Delft University of Technology, 2003. ISBN 90-6464-993-6.

[Pouwelse et al., 2004] J. A. Pouwelse, J. R. Taal, R. L. Lagendijk, D. H. J. Epema, and H. J. Sips. "Real-time video delivery using peer-to-peer bartering networks and multiple description coding." In *Proceedings to IEEE Symposium on Systems, Man and Cybernetics*. The Hague, The Netherlands, 2004.

[Procter, 2000] P. Procter, (ed.) *Cambridge International Dictionary of English*. Cambridge University Press, 2000.

[Ptolemy, 1995–2002] Ptolemy. "The Ptolemy project." www.ptolemy.eecs.berkeley.edu/, 1995–2002.

[Rabaey, 2001] J. M. Rabaey. "Wireless beyond the third generation-facing the energy challenge." In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pp. 1–3. ISLPED'01, ACM, New York, NY, USA, 2001.

[Rajkumar et al., 1997] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. "A resource allocation

model for QoS management." In *IEEE Real-Time Systems Symposium*, pp. 298–307. 1997.

[Rathgeb, 1991] E. Rathgeb. "Modeling and performance comparison of policing mechanisms for ATM networks." *IEEE Journal on Selected Areas in Communications*, vol. 9(3):pp. 325–334, apr 1991.

[Rechtin and Maier, 1997] E. Rechtin and M. W. Maier. *The Art of Systems Architecting*. CRC, 1997.

[van Reeuwijk et al., 2001] C. van Reeuwijk, F. Kuijlman, and H. Sips. "Spar: a set of extensions to java for scientific computation." In *In Proceedings of the ACM 2001 Java Grande/ISCOPE Conference*. 2001.

[Remondo and Niemegeers, 2003] D. Remondo and I. G. Niemegeers. "Ad hoc networking in future wireless communications." *Computer Communications*, vol. 26:pp. 36–40, 2003.

[RM-ODP-10746, 1998] RM-ODP-10746. *Information technology Open Distributed Processing reference model*. International organization for standardization (ISO) and International Electrotechnical Commission (EIC), Dec. 1998. ISO/IEC 10746.

[Roscoe, 1997] B. Roscoe. *The theory and practice of concurrency*. Prentice Hall series in computer science. Prentice Hall, 1997.

[Rosenman and Gero, 1985] M. Rosenman and J. Gero. "Reducing the Pareto optimal set in multicriteria optimization." *Engineering Optimization*, vol. 8:pp. 189–206, 1985.

[Russell, 1989] J. R. Russell. "Full abstraction for nondeterministic dataflow networks." In *30th Annual Symposium on Foundations of Computer Science*, pp. 170–5. 1989.

[Russell, 1990] J. R. Russell. "On oraclizable networks and Kahn's principle." In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pp. 320–8. 1990.

[Saaty, 1990] T. L. Saaty. "How to make a decision: The analytic hierarchy process." *European Journal of Operational Research*, vol. 48:pp. 9–26, 1990. Special issue: Decision making by the Analytic Hierarchy process: theory and applications.

[Salhi et al., 1996] A. Salhi, H. Glaser, D. D. Roure, and J. Putney. "The prisoners' dilemma revisited." Tech. Rep. DSSE-TR-96-2, Department of Electronics and Computer Science, University of Southampton, 1996.

[Satyanarayanan, 2001] M. Satyanarayanan. "Pervasive computing: vision and challenges." *IEEE Personal Communications*, vol. 8(4), 2001.

[van der Schaaf, 1999] A. van der Schaaf. "The ubicom system focus point." Tech. rep., Ubiquitous Communications, 1999. Proceeding of the Ubiconference 99.

[van der Schaaf et al., 2000] A. van der Schaaf, J. A. L. Arellano, and R. I. L. Lagendijk. "On balancing multiple video streams with distributed QoS control in mobile communications." In *2nd Int. Symposium on Mobile Multimedia Systems & Applications (MMSA'2000)*. 2000.

[van der Schaaf and Lagendijk, 2000] A. van der Schaaf and R. Lagendijk. "Independence of source and channel coding for progressive image and video data in mobile communications." *Visual Communications and Image Processing (VCIP2000)*, Jun. 2000.

[van der Schaaf et al., 2001] A. van der Schaaf, K. Langendoen, and R. Lagendijk. "Design of an adaptive interface between video compression and transmission protocols for mobile communications." In *11th Packet Video Workshop (PV-2001)*. Kyongju, Korea, 2001.

[Schmidt, 2002] D. C. Schmidt. "Middleware for real-time and embedded systems." *Communications of the ACM*, vol. 45(6):pp. 43–48, 2002.

[Schreiber et al., 2000] R. Schreiber, S. Aditya, B. R. Rau, V. Kathail, S. Mahlke, S. Abraham, and G. Snider. "High-level synthesis of nonprogrammable hardware accelerators." Tech. Rep. HPL-2000-31, HP Laboratories Palo Alto, May 2000.

[Shannon, 1948] C. Shannon. "A mathematical theory of communication." *Bell Systems Technical Journal*, vol. 27:pp. 379–423, 623–656, 1948.

[Sobieszczanski et al., 1998] S. J. Sobieszczanski, J. Agte, and R. J. Sandusky. "Bi-level integrated

system synthesis (BLISS).” Tech. Rep. NASA/TM-1998-208715, National Aeronautics and Space Administration, 1998.

[Staehli et al., 1995] R. Staehli, J. Walpole, and D. Maier. “A quality-of-service specification for multimedia presentations.” *Multimedia Systems*, vol. 3:pp. 251–63, 1995.

[Steenkiste, 1999] P. Steenkiste. “Adaptation models for network-aware distributed computations.” In *Proceeding of the 3rd Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing (CANPC’99)*. 1999.

[Steere et al., 2000] D. Steere, M. Shor, A. Goel, J. Walpole, and C. Pu. “Control and modeling issues in computer operating systems: resource management for real-rate computer applications.” In *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 3, pp. 2212–2221. 2000.

[Stefanov et al., 2002] T. Stefanov, B. Kienhuis, and E. Deprettere. “Algorithmic transformation techniques for efficient exploration of alternative application instances.” In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign (CODES 2000)*, pp. 7–12. 2002.

[Stefanov et al., 2004] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, and E. Deprettere. “System design using Kahn process networks: the Compaan/Laura approach.” In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, pp. 340–345. 2004.

[Stoilov and Stoilova, 1999] T. Stoilov and K. Stoilova. “Real time noniterative coordination in multilevel systems.” In *Proceedings of Symposium on Large Scale Systems. Theory and Applications (LSS’98)*, pp. 447–452. 1999.

[Strehl et al., 2001] K. Strehl, L. Thiele, M. Gries, D. Ziegenbein, R. Ernst, and J. Teich. “Funstate-an internal design representation for codesign.” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9:pp. 524–44, 2001.

[Stuhlmüller et al., 2000] K. Stuhlmüller, N. Farber, M. Link, and B. Girod. “Analysis of video transmission over lossy channels.” In *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1012–1032. 2000.

[Taal et al., 2003] J. Taal, I. Haratcherev, K. Langendoen, and R. Lagendijk. “Quality of service controlled adaptive video coding over IEEE 802.11 wireless links.” In *ICME, special session on Networked Video*. Baltimore, MD, 2003.

[Taal et al., 2002] J. Taal, K. Langendoen, A. van der Schaaf, H. van Dijk, and R. Lagendijk. “Adaptive end-to-end optimization of mobile video streaming using QoS negotiation.” In *ISCAS, special session on Multimedia over Wireless Networks*, vol. I, pp. 53–56. Scottsdale, AZ, 2002.

[Tappeta et al., 2000] R. V. Tappeta, J. E. Renaud, and J. F. Rodriguez. “An interactive multiobjective optimization design strategy for multidisciplinary systems.” In *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 41st, Atlanta, GA*, AIAA-2000-1665. 2000.

[Tasic and Serdijn, 2002a] A. Tasic and W. Serdijn. “Concept of spectrum-signal transformation.” In *Proceedings IEEE International Symposium on Circuits and Systems*. ISCAS 2002, 2002a.

[Tasic and Serdijn, 2002b] A. Tasic and W. A. Serdijn. “K-Rail diagrams: Comprehensive tool for full performance characterization of voltage-controlled oscillators.” In *Proceedings of International Conference on Electronics, Circuits, and Systems*. ISECS 2002, 2002b.

[Tripathi, 2002] A. Tripathi. “Challenges designing next-generation middleware systems.” *Communications of the ACM*, vol. 45(6):pp. 39–42, 2002.

[Verdu, 1998] S. Verdu. “Fifty years of Shannon theory.” *IEEE Transactions on Information Theory*, vol. 44(6):pp. 2057–2078, 1998.

[Verhoeven and van den Bos, 2001] C. Verhoeven and C. van den Bos. “Reconfiguration of radio receiver front-ends in mobile communications.” In *Proceedings of the 8th Symposium on Communications and Vehicular Technology*. IEEE Benelux SCVT 2001, 2001.

[VRML-97, 1997] VRML-97. *Virtual Reality Model Language (VRML97) International Standard*. The VRML Consortium, 1997. ISO/IEC 14772-1:1997.

[Walpole et al., 1999] J. Walpole, C. Krasic, L. Ling, D. Maier, C. Pu, D. McNamee, and D. Steere. "Quality of service semantics for multimedia database systems." In *Proceedings of the eight working conference on Database Semantics: Semantic Issues in Multimedia Systems*, pp. 393–412. 1999.

[Wan et al., 2001] M. Wan, Z. Hui, V. George, M. Benes, A. Abnous, V. Prabhu, and J. Rabaey. "Design methodology of a low-energy reconfigurable single-chip dsp system." *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 28:pp. 47–61, 2001.

[Wand et al., 1999] Y. Wand, V. C. Storey, and R. Weber. "An ontological analysis of the relationship construct in conceptual modeling." *ACM Transactions on Database Systems*, vol. 24(4):pp. 494–528, Dec. 1999.

[Wand and Weber, 1990] Y. Wand and R. Weber. "An ontological model of an information system." *IEEE Transactions on Software Engineering*, vol. 16(11):pp. 1282–1292, Nov. 1990.

[Wand and Weber, 1995] Y. Wand and R. Weber. "On the deep structure of information systems." *Information Systems Journal*, vol. 5(3):pp. 203–23, Jul. 1995.

[West and Schwan, 2001] R. West and K. Schwan. "Quality events: A flexible mechanism for quality of service management." In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*. 2001.

[Winkowski, 1998] J. Winkowski. "Processes of contextual nets and their characteristics." *Fundamenta Informaticae*, vol. 36:pp. 71–101, 1998.

[Winskel, 1993] G. Winskel. *The formal semantics of programming languages: an introduction*. MIT Press, Cambridge, Mass., 1993.

[Wittgenstein, 1963] L. Wittgenstein. *Philosophical investigations (Philosophische Untersuchungen)*. Blackwell and Mott, Oxford, 1963, 2nd edn. Translation from German by Anscombe, G.E.M.

[Wuu, 1993] S. Wuu, T.-Y.; Vrudhula. "A design of a fast and area efficient multi-input Muller C-element." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1(2):pp. 215–219, 1993.

[Yuan and Nahrstedt, 2001] W. Yuan and K. Nahrstedt. "A middleware framework coordinating processor/power resource management for multimedia applications." In *Proceedings of IEEE Globecom 2001*. 2001.

[Zhao et al., 2002] F. Zhao, J. Shin, and J. Reich. "Information-driven dynamic sensor collaboration." *IEEE Signal Processing Magazine*, vol. 19(2):pp. 61–72, 2002.

[Ziegenbein et al., 2002] D. Ziegenbein, K. Richter, R. Ernst, L. Thiele, and J. Teich. "SPI – a system model for heterogeneously specified embedded systems." *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 10(4):pp. 379–389, 2002.

[Zitzler and Thiele, 1999] E. Zitzler and L. Thiele. "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach." *IEEE Transactions on Evolutionary Computation*, vol. 3(4):p. 257, 1999.

[Zlatanova, 2001] S. Zlatanova. "3D modelling for augmented reality." In *Proceedings of the 3rd ISPRS Workshop on Dynamic and Multi-Dimensional GIS*, pp. 415–420. 2001.

Index of citations

Pasman and Jansen [2001], 26, 31, 55, 58, 133, 153
Pasman and Jansen [2002], 26, 153
Pasman [1999], 133, 153
Patterson and Hennessy [1990], 38, 153
Persa and Jonker [2000], 23, 153
Peterson [1981], 82, 153
Pouwelse et al. [1999], 29, 30, 153
Pouwelse et al. [2000], 117, 153
Pouwelse et al. [2001], 32, 153
Pouwelse et al. [2004], 143, 153
Pouwelse [2003], 144, 153
Procter [2000], 4, 153
Ptolemy [1995–2002], 86, 90, 153
RM-ODP-10746 [1998], 13, 14, 18, 154
Rabaey [2001], 88, 153
Rajkumar et al. [1997], 48, 49, 153
Rathgeb [1991], 59, 154
Rechtin and Maier [1997], 10, 154
Remondo and Niemegeers [2003], 143, 154
Roscoe [1997], 82, 87, 88, 154
Rosenman and Gero [1985], 62, 154
Russell [1989], 91, 95, 154
Russell [1990], 83, 90, 154
Saaty [1990], 31, 43, 66, 154
Salhi et al. [1996], 56, 154
Satyanarayanan [2001], 58, 154
Schmidt [2002], 48, 154
Schreiber et al. [2000], 65, 154
Shannon [1948], 50, 53, 154
Sobieszczanski et al. [1998], 66, 154
Staehli et al. [1995], 48, 155
Steenkiste [1999], 46, 47, 155
Steere et al. [2000], 47, 155
Stefanov et al. [2002], 85, 155
Stefanov et al. [2004], 144, 155
Stoilov and Stoilova [1999], 65, 155
Strehl et al. [2001], 84, 155
Stuhlmüller et al. [2000], 127, 128, 155
Taal et al. [2002], 11, 44, 47, 58, 155
Taal et al. [2003], 144, 155
Tappeta et al. [2000], 66, 155
Tasic and Serdijn [2002a], 133, 155
Tasic and Serdijn [2002b], 29, 133, 155
Tripathi [2002], 48, 155
VRML-97 [1997], 25, 155
Verdu [1998], 50, 155
Verhoeven and van den Bos [2001], 28, 34, 155
Walpole et al. [1999], 48, 156
Wan et al. [2001], 88, 156
Wand and Weber [1990], 14, 18, 156
Wand and Weber [1995], 18, 19, 156
Wand et al. [1999], 18, 19, 156
West and Schwan [2001], 49, 83, 156
Winkowski [1998], 82, 156
Winskel [1993], 87, 88, 156
Wittgenstein [1963], 57, 156
Wuu [1993], 96, 156
Yuan and Nahrstedt [2001], 37, 156
Zhao et al. [2002], 59, 156
Ziegenbein et al. [2002], 84, 90, 156
Zitzler and Thiele [1999], 65, 71, 156
Zlatanova [2001], 26, 156
de Bruin and Nienhuys-Cheng [1998], 89, 147
de Kock et al. [2000], 83, 90, 98, 150
van Dijk and van Reeuwijk [1999], 29, 30, 115, 147
van Dijk et al. [2000a], 11, 58, 147
van Dijk et al. [2000b], 11, 125, 147
van Dijk et al. [2003], 12, 147
van Gemund [1996], 33, 88, 89, 148
van Reeuwijk et al. [2001], 33, 154
van den Bos and Verhoeven [2001], 132, 146
van den Bos [1999], 34, 146
van der Aalst [1999], 19, 145
van der Schaaf and Lagendijk [2000], 120, 154
van der Schaaf et al. [2000], 69, 154
van der Schaaf et al. [2001], 126, 139, 154
van der Schaaf [1999], 4, 154
AOSD-web [2001–2002], 48, 145
Boijmans Van Beuningen [1849,1958], 2, 146, 148
Coello Coello and Christiansen [1999], 65, 147

Samenvatting

De trend in communicatiesystemen is om steeds meer functionaliteit in steeds kleinere apparaten te stoppen. Een steeds groter wordend deel van de functionaliteit moet daarbij ook nog eens inspelen op veranderingen in de omgeving en in de gemoedstoestand van de gebruiker; de applicaties moeten zogezegd context-aware zijn. Met de huidige stand van de techniek gaat dit niet lukken, al was het alleen maar vanwege het beperkte budget aan beschikbare middelen dat dergelijke apparaatjes bij zich kunnen dragen. Echter het functierepertoire van moderne mobiele communicatiesystemen is groter dan de functionaliteit die tegelijkertijd beschikbaar moet zijn, bovendien kunnen functies op verschillende manieren worden uitgevoerd. Het gevolg is dat communicatiesystemen in een voortdurende staat van verandering zijn (in flux). Deze dissertatie beschrijft een ontwikkelmodel om de samenwerking tussen functies te coördineren, gegeven het beperkte budget aan beschikbare middelen, maar gebruikmakend van het brede functierepertoire. De rode draad in deze dissertatie is het Ubicom systeem, een communicatiesysteem dat de werkelijkheid van de gebruiker verrijkt met animaties.

Probleem

Complexe systemen worden gekenmerkt door een veelvoud van disciplines die nodig zijn voor de ontwikkeling van dergelijke systemen. Bovendien zijn dergelijke systemen in een voortdurende staat van verandering. Een modern systeem wordt geacht instantaan te reageren op wisselende omgevingscondities en/of veranderende eisen van haar gebruikers.

Het opdelen van een systeem in componenten is een beproefde methode van complexiteitsbeheersing. Dit stelt echter hoge eisen aan de coördinatie van diverse componenten. Coördinatie is de lijm die van losse componenten weer een systeem maakt. Er bestaan goede manieren om grote homogene systemen te ontwikkelen. Dergelijke manieren maken gebruik van de homogeniteit om de complexiteit te beheersen, meestal met behulp van een centrale controller. Niet homogene, zogenaamde heterogene, systemen vragen echter een oplossing zonder centrale controller, onder andere vanwege het feit dat de schaalbaarheid van centraal beheerde oplossingen te wensen overlaat.

In deze dissertatie beschouwen we een gedistribueerde aanpak, waarin alle componenten tezamen verantwoordelijk zijn voor de kwaliteit van het systeem. De centrale

vraag is dus hoe de coödinatie tussen de componenten effectief kan worden geregeld. We zoeken nadrukkelijk naar een flexibel ontwikkelmodel. Dit model moet niet alleen voldoen aan de eis van het systeem om context-aware te zijn, we willen ook de mogelijkheid hebben om nieuwe ontwikkelingen op enig gebied van de betrokken disciplines eenvoudig op te kunnen nemen.

Resultaat

Deze dissertatie heeft drie pijlers: aspect specifieke beschrijving van communicerende systemen, een onderhandelingsmodel ten behoeve van de coördinatie en de noodzaak om uiteindelijk compositie te kunnen plegen.

Systemen zijn concrete "dingen" met concrete eigenschappen. De waardering van een eigenschap is echter context afhankelijk. Het hangt ervan af hoe je een systeem benadert en hoe je een systeem in zijn omgeving plaatst. Om grip op de zaak te krijgen introduceren we het ontologiemodel van Bunge. Dit model biedt de mogelijkheid om systemen te beschrijven vanuit verschillende perspectieven, waarbij de relaties tussen de beschrijvingen onderling expliciet worden gemaakt. We gebruiken het Ubicom systeem om de verschillende beschrijvingen te illustreren. Het gebruik van specifieke beschrijvingen maakt dat een component een consistent beeld kan verkrijgen van zijn omgeving (context) zonder dat het nodig is daarvoor een meta-model te ontwikkelen. Merk op dat naburige componenten overlappende context hebben.

Bovenstaande observaties zijn geïmplementeerd in een onderhandelingsmodel: het ARC model. Dit model berust op drie begrippen: abstractie, adaptatie, en coöperatie. Componenten die het ARC model ondersteunen gebruiken een model (abstractie) om hun omgevingskenmerken te beschrijven: verstoring, capaciteit en gebruik van middelen. Met dit model kunnen componenten communiceren met naburige componenten om een beeld te krijgen (adaptatie) – en te geven (coöperatie) – van de omgeving waarin ze opereren. De communicatie verloopt via een drie-staps protocol: vraag, aanbod en contract. Het uiteindelijke contract is typisch het resultaat van een zogenoemd multi-objective optimalisatie probleem. Vanwege het feit dat een aanbod meerdere opties bevat hoeft er niet eindeloos te worden onderhandeld

Een specifieke implementatie van het ARC model wordt gedaan in ons CAPN rekenmodel. In dit model beschouwen we datastromen die zo nu en dan moeten reageren op externe gebeurtenissen. CAPN is in beginsel een niet deterministisch rekenmodel, echter het model kan worden gemodelleerd in een rekenmodel dat wel deterministisch, het Kahn model. Deze modellering levert een expliciete relatie op tussen het systeem en zijn context. In specifieke gevallen kunnen we een symbolische expressie afleiden die deze relatie beschrijft. De expliciete contextrelatie maakt het mogelijk om op een deterministische wijze compositie te plegen.

Acknowledgement

This dissertation is the result of research conducted in the Ubicom programme. In the dissertation I emphasise the multidisciplinary character of the project in more than one way. For one, I write "we" instead of "I", because I feel this research is the result of joint work.

I am in depth to many people. I thank Prof. Ed Deprettere for inviting me to return to Delft University of Technology after a few years in industry. I thank the Ubicom project management team for offering me the position of system architect and giving the opportunity to develop the position in an unforeseen interesting direction. The wide scope of the project taught me many valuable things, for one thing: you don't always have to be in the right.

Prof. Henk Sips and Prof. Inald Lagendijk had a major positive influence on this dissertation as my promotors. In our many discussions they insisted on being precise, focused, and concrete. Initially we faced an exponentially growing multi-something problem: multidisciplinary, multilevel multi-optimisation, etc. Gradually we got a grip on the problem. I think we succeeded.

The ARC framework described in this dissertation is joined work with Dr. Koen Langendoen. The validation and finesse though, is due to many invaluable discussions I had with all researchers, Ph.D students, and graduate students involved in the Ubicom project. The Ubicom community was open minded and willing to think, act, and contribute. Thanks!

The CAPN model of computation would not have been developed without the involvement of Prof. Ed Deprettere, Ir. Paul Lieverse, and Dr. Arjen van Gemund.

I unscrupulously borrowed ideas from the many vivid discussions I had with my (part-time) roommates and neighbours: Dr. Arjan van der Schaaf, Ir. Maarten Ditzel, Ir. Jacco Taal, Ir. Johan Pouwelse, Dr. Wouter Pasman, and Dr. Koen Langendoen.

Finally I thank my wife and children for their (im)patience and understanding; "sit Heit alwer yn syn hok?".

Hylke W. van Dijk,
July 4th, 2004,
Katlijk.

Curriculum Vitae

Hylke W. van Dijk was born in Wommels (1965). He received an ing. degree in ships engineering (1987), but set sail as a programmer of educational software for the next two years. In 1993, Hylke received an Ir. degree from the network theory (Electrical Engineering) section of Delft University of Technology. He continued to work on parallel implementations of Jacobi-type algorithms (SVD, EVD, RLS, etc.). In 1996, Hylke joined Philips to educate customers about the potentials of an high-end multi media processor: TriMedia. In 1998 he returned to Delft University of Technology as a system architect for the Ubicom programme; a multidisciplinary research program aiming at mobile visual augmented reality. Currently, Hylke is postdoc in the Software Engineering group of Delft University of Technology; still working on complex embedded systems their design, evaluation, and integration.