Architectural Design

Chris A. Vissers · Luís Ferreira Pires
Dick A.C. Quartel · Marten van Sinderen

Architectural Design

Conception and Specification of Interactive Systems

Springer

Chris A. Vissers
University of Twente
Enschede
The Netherlands

Dick A.C. Quartel
BiZZdesign
Enschede
The Netherlands

Luís Ferreira Pires
University of Twente
Enschede
The Netherlands

Marten van Sinderen
University of Twente
Enschede
The Netherlands

ISBN 978-3-319-43297-7 ISBN 978-3-319-43298-4 (eBook)
DOI 10.1007/978-3-319-43298-4

Library of Congress Control Number: 2016947019

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Preface

This book is entitled *Architectural Design—Conception and Specification of Interactive Systems*. What do we mean by this and what is the purpose of this book?

Architectural Design

By *architectural design* we mean the design of the functional behaviour of a system, and the design of the internal structure of the system as a composition of (high level) functional units. This means that this book introduces a *design methodology* that starts with and remains close to the requirements of the end users of the system. At this so-called *architectural level*, we are not yet concerned with detailing and defining the low-level mechanisms that implement and perform this functional behaviour, which should happen in later design steps. However, some general knowledge of the *implementation level*, which is often quite specific for various types of systems, may appear supportive for understanding this book. For ICT systems, for example, we consider the choice of detailed algorithms, software programs and hardware units as implementation concerns.

Amongst the target systems we aim at are ICT systems, such as large software systems and various process-oriented systems in business, production, organisations and administrations.

Conception and Specification

In our design methodology, we make a sharp distinction between the *conception of a design* and the *specification of a design*.

We consider the *conception of a design* as an intellectual process that takes place in the mind of designers in which a design is created as a composition of *conceptual (functional) building blocks*. If these building blocks are close to the intuitive understanding of the designers, they contribute positively to the insight in and overview of what is being conceived, and thus to the ability of these designers to master the design process. The availability of building blocks that can be effectively applied in the design of a broad range of possibly complex systems is of particular interest for a methodology that focuses on the design of these systems. We call these conceptual functional building blocks as *basic design concepts* in this book and they form the basic constituent of our design methodology. The *interaction* concept and the *causality relation* concept are examples of basic design concepts.

We consider the *specification of a design* as a human and machine interpretable representation of a conception and an inherent and indispensable complement to the design itself. The specification of a design plays four roles in our design methodology:

- As a window through which a designer can view a conception and get better grips on what is being conceived.
- As a communication means between the designer and the end user of a system, while discussing and possibly reformulating the user requirements for the system.
- As a communication means amongst designers and design groups, while refining a design in further design steps.
- As a basis for verification, correctness preserving transformations towards implementation and software tool development.

To perform its communication roles effectively, the specification language should posses ‘expressive power’. This means that designers should be able to express their basic design concepts directly and concisely in the language, so that these concepts can be unambiguously recognised. In contrast, the designer should not be forced to express a basic design concept by an unwieldy and relatively arbitrary composition of too elementary language elements, where neither the language elements nor their composition have a direct relationship to the basic design concept. Expressive power allows the hand-in-hand application of a conception and its specification in a seamless way. We consider the latter as essential in *architectural design*.

To properly perform all the above-mentioned roles, the specification language should have a concrete syntax and formal semantics, so that precision is guaranteed and ambiguity is excluded.

The specification language that we present in this book has been devised to allow the direct representation of our basic design concepts. In this way, we can focus on architectural design and we avoid language constructs that are only relevant at implementation level.

When using the term *design*, we generally mean the hand-in-hand *conception and specification* of a design.

Interactive Systems

A system that does not interact with its environment is quite useless and as such it should not be designed. In this respect, our use of the term *Interactive* in the title sounds like a tautology, since useful systems always interact.

Apart from considering systems in general, however, we have a special interest in the interactions between systems[1] that, together, form a total system. Usually, when considering interactive systems, we are inclined to first focus on the systems as individual objects and only in second instance consider their interactions as additional phenomena. However, we also know from practice that mechanisms such as protocols and interfaces strongly influence the structure and functioning of a system in its totality. This raises several questions as follows:

– Which explicit functional goal do these interactions aim to achieve in the functioning of the system in its totality?
– How can this functional goal be recognised and designed?
– Can we use the design of this functional goal as a building block in the design of the system consisting of the interactive systems?
– What are the merits of designing a system this way?

Similar questions apply to the interactions, both at a high and low functional level, between the subsystems that are internal to a system and together form this system. Answers to these questions are highly relevant, since interacting systems and subsystems appear at very large scale in the fields of engineering, organisation and administration. In this book we discuss the *Interaction System* concept as a basic design concept that provides such answers.

We consider *distributed systems* as important representatives of interactive systems. Examples are business processes, production systems and ICT systems, such as the Internet and mobile phone systems.

Design Methodology

In this book we present a design methodology that is practically applicable to the architectural design of a broad range of systems in various fields of discipline.

In the first instance, it enables the system architect to assist the user in choosing and defining appropriate functional requirements for the system in its totality, and specify them in their most precise, concise, surveyable and understandable way. In the second instance, it enables the designer to devise the internal structure of the system, i.e. as a composition of subsystems, in increasingly more detail, until a structure is obtained that can act as a prescription for the implementation of the system.

Our design methodology is based on design concepts with a basic and fundamental nature that are not susceptible to ageing or fashion, proving long-lasting applicability. The concepts are independent of specific functions and technologies that can be chosen to eventually implement a concrete system. This implies that we do not focus on these choices nor advocate for them.

[1]Seen from the outside, a human being that interacts with a system acts just as another system. This implies that HCI (Human Computer Interfacing or Human Computer Interaction) is implicitly covered by our approach, although it is not an explicit point of attention in this book.

To facilitate the understanding of our concepts and methods, we provide many tangible, appealing and easy-to-recognise examples from various fields. We think that recognising and understanding these examples not only provides eye-opening insights, but is also fun. The examples can be often related to ICT problems, showing that we can often treat ICT and non-ICT problems with a coherent approach. In these examples, certain specific functional choices necessarily have to be made, but these choices are only meant for illustrative purposes.

Applicability of the Methodology

The main condition for the applicability of our methodology is that the target system can be properly represented with our concepts. This is particularly true for systems where the dynamic part of their behaviour, i.e. the mutual dependency and sequencing of discrete interactions, is dominant. Since this is the case for many types of systems, our methodology is applicable and has been effectively applied to a large variety of systems.

In the presentation of our methodology we spend marginal attention to methods where the represention by discrete values that are established in interactions, the ontological relationships between these values, the storage and retrieval of large volumes of such representations, the integrity of these representations, and the operations on them are the dominant factors. However, our methodology can in principle be linked to such methods.

The work is not applicable to fields where the design concepts cannot properly be represented by interactions, for example, when these concepts come close to low-level software and hardware engineering or the monitoring and control of continuous values.

Target Audience

The target audience of this book consists of professionals, practitioners, managers and administrators in industry and large organisations who are responsible for design, development, installation, testing, maintenance, extension, management, supervision and control of large and complex systems. We also aim at students in graduate courses who want to develop professional insights and skills in developing complex systems. For this purpose, we paid special attention to the didactics in the text. Earlier versions of this text have indeed been used as lecture notes in courses on services, protocols and interfaces presented at the University of Twente. This implies that the book can be used as a textbook in graduate courses.

Brief History

Our insights in design methodology came forward out of research in distributed (ICT or Telematics) systems in general. This research has been carried out at the University of Twente, the Netherlands, and was started back in 1967. Therefore, our methodology builds on a long tradition and rich history of original work. In 1992, the Telematics Institute (one of the four Dutch national top technological institutes) joined in this research.

Around 1992 we observed that contemporary techniques, such as the Formal Specification Methods (FMSs) CSS, CSP, SDL, Petri Nets and LOTOS too often forced a designer to conceive and specify a system by defining unwieldy compositions of very elementary language primitives. Some of these techniques appear even averse from engineering practice, and they force a designer to think more in terms of a mathematical theory rather than providing a focus on practical design. This formed the background for our ambition to strive for more pragmatic, engineering-oriented and intuitively appealing design constructs with direct and high-design capabilities, yet without compromising precision and unambiguity. This work resulted in the design methodology presented in this book.

This research has led to several publications, of which we mention three Ph.D. theses in particular because they first introduced the original insights, concepts and motivation for our design methodology: the Ph.D. thesis of Chris A. Vissers, ‘Interface, a dispersed Architecture’ (1977); the Ph.D. thesis of Luís Ferreira Pires, ‘Architectural Notes: a Framework for Distributed Systems Development’ (1994); and the Ph.D. thesis of Dick A.C. Quartel, ‘Action relations, basic design concepts for behaviour modelling and refinement’ (1998).

Our research has also led to many contributions to international conferences, large-scale European projects, periodicals and standardisation committees.

Industrial Impact

The ideas and concepts presented in this book formed the inspiration and basis of two large language and software tool development projects: Testbed (1996–2001) and ArchiMate (2002–2004). Both projects were carried out by the Telematics Institute, Enschede, the Netherlands, and involved several universities and large organisations. The result of Testbed was a model-based test environment for the analysis, improvement and redesign of business processes in (large) organisations. This environment consisted of a process modelling language, called Amber, supported with methods and techniques and an extensive toolset. A company called BiZZdesign was founded in 2001 as a spin-off of the Testbed project, and this company turned this environment into a successful product in the Business Process Management market, branded under the name BiZZdesigner. The main result of ArchiMate was a language for modelling the architecture of enterprises. An enterprise architecture typically describes (the relationships among) the products and services of an organisation, the business processes that realise these products and services, the software applications that support these processes, and the infrastructure on which these applications are deployed.

ArchiMate has become an international standard in 2009, and its development is fostered by the ArchiMate forum of The Open Group. Version 2.1 of the language was published in 2013. The language is now supported by many tool vendors, among which BiZZdesign, who was the first to offer a native and user-friendly implementation of a tools suite to support ArchiMate, called BiZZdesign Architect. This implementation supports various powerful analysis techniques in addition to modelling. With the products BiZZdesigner and BiZZdesign Architect, BiZZdesign has become a major player in the areas of Business Process Management and Enterprise Architecture, and now employs more than 100 people worldwide.

Reading Guidelines

The difficulty in reading this text may come mainly from the several concepts that at first sight may appear artificial, sophisticated and abstract. The precise definition we choose for these concepts may add another dimension to this difficulty. Abstraction and precision, however, are the indispensable attributes for understanding complex systems and precisely conceiving and representing them at a high functional level. Once understood, these concepts only appear as natural, self-evident and extremely powerful, because they can reflect directly, precisely and concisely what is considered essential in the functional behaviour of a system, i.e. they emerge as eminent *architectural* concepts.

Chapters 1 and 2 present our global views on how to design systems and how to interpret terms and meta-concepts that are frequently used in design and design specification approaches. These chapters are introductory and informal in nature, and provide the general context in which the remaining chapters can be read.

Chapters 3 through 6 present most of our basic design concepts, and illustrate them with examples. Language notations are introduced along with the basic design concepts. These chapters are formal in nature and more difficult to read. After fully mastering the material of these chapters, the reader should be capable of designing an arbitrarily complex system, both as a totality and as a composition of subsystems.

Chapters 7 through 12 discuss the more intricate basic design concept of *interaction system*, which forms the core of many *interactive systems* by focusing on their *common functional goal*. These chapters are recommended to readers who have a particular interest in the design of protocols and interfaces for various systems. The chapters use the concepts introduced in Chaps. 3 through 6. Examples are predominantly taken from ICT systems.

Chapter 7 elaborates on the *interaction system* concept, leading to a particular view on the notion of *service* and *protocol*, where a *protocol* implements a *service*. A global design approach for interaction systems leads to the notions of *separation of concerns* and *layered architectures*. Some well-known instances of practical interaction systems are shown as examples.

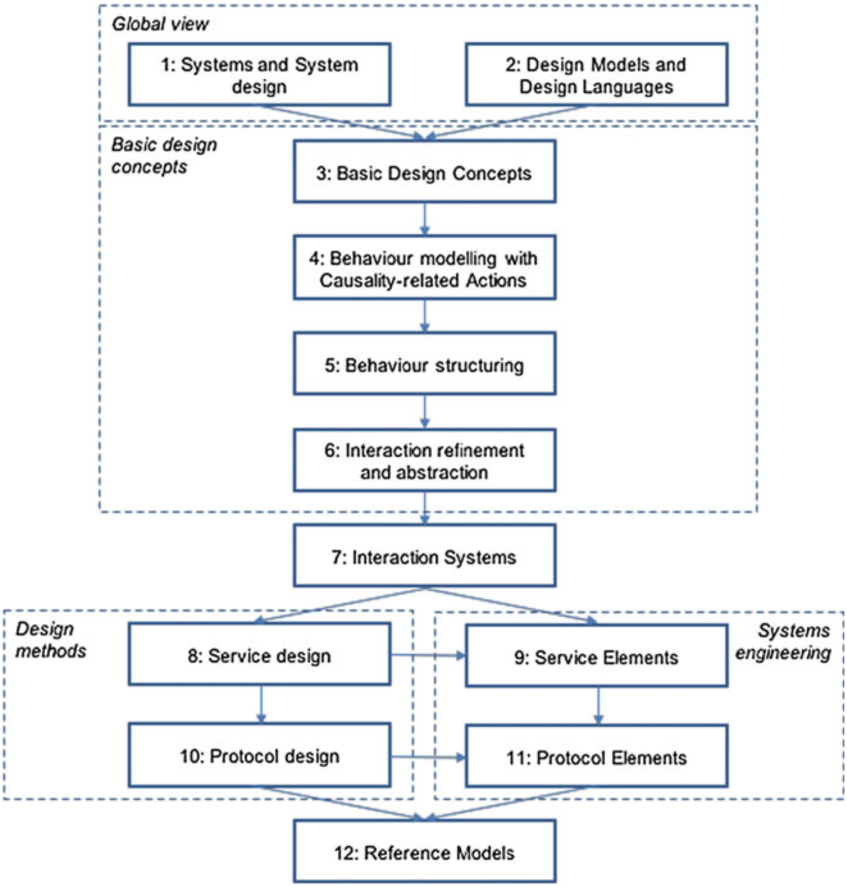
Chapter 8 presents a generally applicable method for structuring a service that allows to control its complexity. The method is based on the *constraint oriented structuring method* introduced in Chap. 5.

Chapter 10 provides a generally applicable method for structuring a protocol that provides insight in the generally high complexity of a protocol and allows to control its design. The method builds further on Chap. 8 and on the notion of separation of concerns.

Chapters 9 and 11 are much more targeted to ICT systems engineering. They present concrete technical functions and their possible relationships that can be frequently encountered in concrete services and protocols. In particular, it shows how *protocol functions* can implement *service functions*.

Chapter 12 discusses the concept of *reference model* as a structure of related services and protocols that together form a complex interaction system. By only specifying the key functions in these services and protocols, a reference model can be defined first and used later to organise the cooperation of different design teams to work concurrently to complete the design of an interaction system. The concept of reference model can mutatis mutandis be used for the design of complex systems in general.

The figure below shows the relationships between the chapters of this book.



Closing Remarks

For reasons of keeping this book coherent, accessible and feasible, we restrict ourselves to only presenting the basic technology-independent principles that underlie our design methodology. This implies that we refrain from entering into or amply referring to the overwhelming amount of contacts, publications, activities, projects, software tool productions and other developments that came forward out of, are inspired by, or are associated with our original work. We trust that these principles, once understood, contribute to essential and proper insights for a better control of the architectural design of systems.

Enschede, The Netherlands
April 2016

Chris A. Vissers
Luís Ferreira Pires
Dick A.C. Quartel
Marten van Sinderen

Contents

List of Definitions