Boderc: Model-based design of high-tech systems

A collaborative research project for multi-disciplinary design analysis of high-tech systems.

Embedded Systems Institute (The Netherlands)

Boderc: Model-based design of high-tech systems

A collaborative research project for multi-disciplinary design analysis of high-tech systems.

Editors:

Maurice Heemels Eindhoven University of Technology
Gerrit Muller Embedded Systems Institute

Publisher:
Embedded Systems Institute, Eindhoven, The Netherlands

Publisher:
Embedded Systems Institute
TU/e Campus, Den Dolech 2
P.O. Box 513, 5600 MB Eindhoven
Eindhoven, The Netherlands

First edition	:	November 2006,	presented at symposium 2006.
Second edition	:	March 2007,	updated version.

Keywords:
system design; system engineering; high-tech systems; modeling; high-level method; performance; multi-disciplinary

ISBN-13: 978-90-78679-01-1
ISBN-10: 90-78679-01-8

© Embedded Systems Institute, Eindhoven, The Netherlands 2006
All rights reserved. Nothing from this book may be reproduced or transmitted in any form or by any means (electronic, photocopying, recording or otherwise) without the prior written permission of the publisher.

The Boderc project has been executed under the responsibility of the Embedded Systems Institute, and is partially supported by the Netherlands Ministry of Economic Affairs under the Senter TS program.

Foreword

Around 2001 Océ Research & Development experienced a paradox regarding the future role of informatics in product design. In those days the world was at the summit of the internet hype. Océ, which is in the business of professional printing and document management, handled these challenges in a prudent way - it was careful not to be diverted from its bread and butter business. However, it did not pass unnoticed that product development became more and more dependent on the results and challenges of embedded software technology, as well as on the effective interplay of software, electrical, and mechanical engineering. Software engineers were exposed to two fundamentally conflicting roles. On the one hand, they had to be aware of the organisational risks that relate to new business development, while the sky seemed the limit in the world of their peers. On the other hand, they were exposed to the daily operational pressure to solve hard engineering problems in embedded systems. So it was felt that, besides facilitating Océ's entrance in the software and services business, there was a significant need to improve the multidisciplinary interaction between the embedded software, electrical and mechanical engineering disciplines, starting from the very early phases of printer development.

It was in this period that the Embedded Systems Institute was scouting for their first project ideas. This helped us to amplify the abovementioned challenges. An analysis of the state of affairs led us to the hypothesis that multidisciplinary design interaction in the earliest project stages, with full involvement of the IT discipline, should create significant value. The possibility to explore this assumption together with the Embedded Systems Institute, was considered a major opportunity.

This was one of the motivations to start the Boderc project, in which Océ participated with research groups from the three technical universities, other industrial partners, and ESI. The shared problem statement was the 'multidisciplinary design of print engines'. A central theme from the start of this project was the role of models in multidisciplinary design, or, alternatively stated, a shared (formal) language across disciplines. Models support the process of sharing knowledge; models allow to perform analyses; models can be applied to validate expected results. Still, it was not clear what information those models should provide and how we should build them. The search for an answer to these questions was considered the main challenge of the Boderc research program.

And now, here we are with the results: a book that summarizes the research findings of the Boderc project. So, what has happened with the project in the context of Océ R&D? In my opinion this project has brought us both useful, specific results, and promising general directions. It is also the start of a much longer lasting process that aims to further pervade our R&D and the other partners. Some results entirely fulfilled our initial expectations, the best known example being the 'Happy Flow' model. This enabled us to skip at least one complete physical machine-build iteration, because pa-

per path designs could now be explored virtually. The savings from this result alone already amount to many man-years of effort. Nevertheless, the process creating such models systematically is not yet very transparent, although the positive influence of having research projects like Boderc, and therefore room for exploration, is crystal clear.

All in all, we are very happy to have embarked upon this collaborative research project. Product development has gained significantly from the results and our engineers have become more effective in the early design phases of new printers. We value our encounter with the Embedded Systems Institute, knowing that Boderc was the first project of this organization, and wish them many more successful innovations. We are confident they will further their impact, pursuing the cross-fertilization between industry and their problems and the fundamental insights of academia. Together we are currently exploring the possibilities for continuing our fruitful cooperation. Learning from each other has to become *the* contributing factor, and we are proud to have taken the first steps towards this goal!

Ir. W. Orbons
Senior Vice President Research & Development
Océ Technologies B.V.
Venlo, The Netherlands
October 2006

Preface

This book is the result of a team effort in the truest sense. It reports on the results of the Boderc project, which was the very first collaborative research project of the Embedded Systems Institute that has been carried out in an 'Industry-as-Laboratory' setting. This is a unique research formula that combines the strengths of industrial companies, universities, and research institutes, with the objective to create breakthrough applied research under realistic industrial constraints.

This book summarizes the key results of the Boderc project and represents the collective work of researchers and engineers from the companies Océ-Technologies, Chess, and Imtech, together with research groups at the Technical University Eindhoven, University of Twente, Radboud University Nijmegen and the Embedded Systems Institute. Representatives from these organizations have worked together for about five years, focusing their research on an industrial problem statement from Océ-technologies. In effect, this book constitutes the accumulated knowledge and experience of this unique multidisciplinary community.

Publishing the project results in this book cannot be done without expressing our gratitude to those who contributed to the success of Boderc. Many have participated in the project, including at least six PhD-student advisors, six university professors, and three industrial managers to guide the work. We would like to express our gratitude to all our partners in both industry and academia, as it was their contribution that enabled the success of this ambitious project. The funding by Océ Technologies and the Dutch Ministry of Economic Affairs provided the essential financial means to carry out the project. We are confident that it has brought significant benefits to the partners and will be a source of inspiration for them, as well as all other interested parties in the future.

Prof. dr. Ed Brinksma
Scientific Director & Chair
Embedded Systems Institute
The Netherlands
October 2006

Contents

1	Introduction	1
2	A design methodology for high-tech systems	11
3	The key driver method	27
4	Threads of reasoning	43
5	Budget-based design	59
6	Effective industrial modeling: The example of Happy Flow	77
7	Heat modeling in copiers	89
8	Modeling of performance	101
9	Virtual printer modeling	115
10	Using stepper motors in printers	129
11	Simulating the environment of embedded software	141
12	Evaluating embedded system architectures	151
13	Model-driven design of real-time systems	161
14	Time-varying delays in control	171
15	Sheet feedback control in a printer paper path	183
16	Event-driven control	193
17	Design trajectory and controller-plant interaction	203

18 Impact, lessons learned and conclusions **213**

A List of Boderc publications **225**

B List of authors **231**

Chapter 1

Introduction

Authors: W.P.M.H. Heemels, G.J. Muller and P.F.A. van den Bosch

The design of high-tech mechatronic systems like wafer steppers, electron microscopes, copiers, et cetera, is a complex process. Multiple 'classical' engineering disciplines need to make the overall design in close co-operation. Typically, electrical, mechanical, software and other engineering disciplines together determine the functioning of the final product. Especially in the early design phases, the design of the product is vulnerable for erroneous design choices as many others will be based on it subsequently. These erroneous choices tend to show up in later phases during the integration or even the manufacturing itself. Late in a project, the 'repairs' are more difficult and can lead to a much longer development period than planned and/or a less optimal product.

The main reasons for non-optimal design choices, of which some are illustrated in Figure 1.1, are summarized below.

- A common language and background between multiple engineering disciplines, which enable the reasoning about system properties, are lacking. As a result, the consequences of a choice, made by one discipline, cannot be overseen for other disciplines; wrong assumptions are made on the sub-designs of other disciplines; confusions and misunderstanding are present about definitions of specific terms and priorities differ over disciplines.

- Many design choices are made in an implicit way, based on experience, intuition and 'gut-feeling.' That way, it is hard to communicate the reasons and to discuss the design or particular choices in it. Decisions are sometimes not well-founded by quantitative arguments, but can be forced by seniority and 'shouting loudest.'

- Especially dynamic, time depending aspects of a system are complex to grasp. There are not many tools and methods available to support the time varying aspects in a design, in contrast to many static or steady-state aspects.

- Out-of-phase project evolution is another important factor. A typical example of the latter is that the mechanical design often precedes the electronic design, which on its turn precedes the software design.

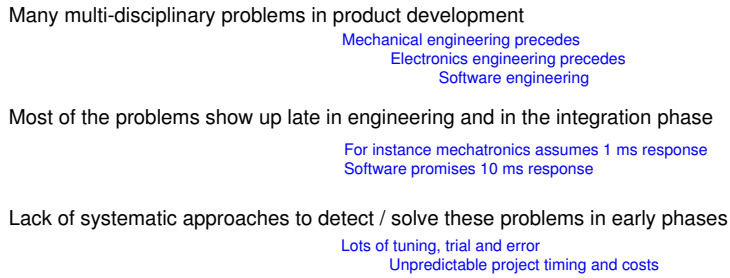


Figure 1.1: Typical industrial problems in mechatronics systems

The effects of the above design complications are amplified by the size and complexity of high-tech machines (typically millions lines of codes, tens of thousands mechanical components like pinches, springs, belts, motors, bolts, et cetera). The more complex the machine and the more people involved in the design, the stronger the effects. Of course, the four mentioned reasons are not the only ones that complicate the design. Other factors like organizational or political, or geographically scattering of the design team contribute too. Those latter issues are important as well, but are of a different dimension and more related to business management. We believe that the aforementioned reasons can be relieved by the use of models that capture the system behavior and a reasoning method that indicates how and when to use them. That is why the Bode-RC project was initiated by the Embedded Systems Institute, Océ Technologies, Imtech, Chess and 6 academic groups of the universities of Eindhoven, Nijmegen and Twente.

1.1 The Boderc project

Early on in the Boderc project, the goal was defined as shown in annotated form in Figure 1.2. The goal of Boderc is to develop a model-based methodology that supports multi-disciplinary design (space exploration) by predicting system performance. The developed models, methods and techniques should in particular be applicable in the early design phases and must satisfy industrial application constraints. They should be usable in the industrial context with its particular people, processes and economic constraints related to design time, effort and costs. Moreover, the economic constraints and the traditional processes of the manufacturer of the product restrict the design space a priori by posing constraints on the design. Most parts in a new design will not be revolutionary, existing solutions and technologies and way of working will be

re-used, which constrains the design space. The methodology should be effective for this constrained design space.

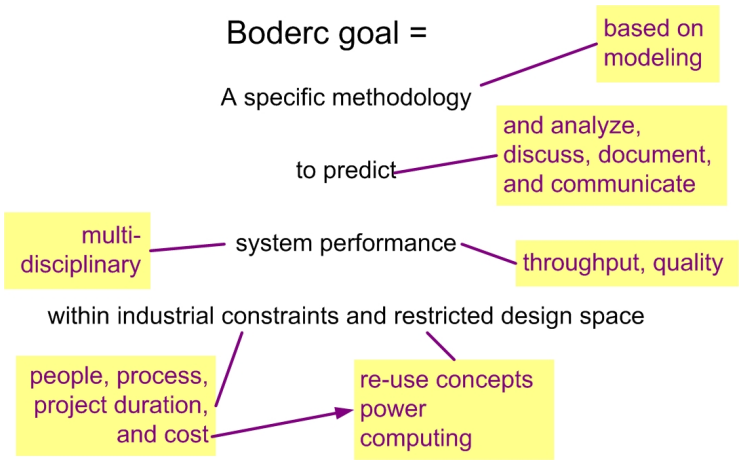


Figure 1.2: Boderc research project goal

During the Boderc project the awareness emerged that it is not only about *predicting* system performance. The methodology and models force to make design choices quantitative and explicit which enables the analysis of various design options, communication between engineers from different disciplines and to commence the design with all disciplines involved in the beginning of a project. Also modeling of (parts of) the system increases the understanding and insight in the design. All these factors lead to shorter design iterations and more confidence in the consequences of design choices. In the end, better products are delivered faster.

Boderc: name and logo

The Boderc name and logo deserve also some explanation, especially considering the various gimmicks being in it. Boderc stands for Beyond the Ordinary: Design of Embedded Real-time Control. In the logo depicted in Figure 1.3 one can observe that the first 4 letters ‘Bode’ are separated from ‘RC.’ Bode refers to the frequency response functions in the form of Bode plots [14], one of the fundamental means to indicate the performance of linear control systems in control theory. The ‘RC’ are capitalized and indicate the letters that stand for resistors and capacitors, important components in many electrical circuits. Also the letter ‘0’ has an integral through it denoting a so-called circle integral, which is well-known in mathematics. This reflects the desirable connection of the Boderc results to scientific foundations (next to industrial applicability). Moreover, in this manner the Boderc logo indicates the multi-disciplinary nature of the project. As Boderc aims at developing a *model*-based design methodology, the

pronunciation of Boderc refers to the actress Bo Derek, who after all is also a kind of model.

Figure 1.3: Boderc logo

Research method: Industry-as-Laboratory

The Boderc project uses the *industry-as-laboratory* approach, as proposed by Colin Potts [94] and visualized in Figure 1.4.

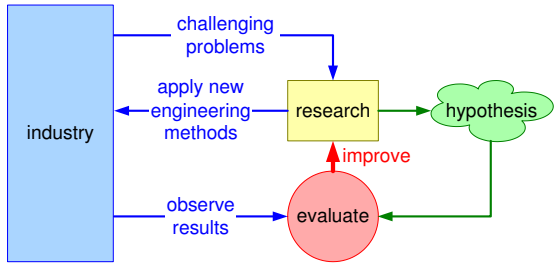


Figure 1.4: Industry as Laboratory: Research of engineering methods

The industry-as-laboratory approach exploits the actual industrial setting as a test environment, which warrants that the research question is based on real industrial problems. The Boderc research team, consisting of a mix of academic and industrial people, investigates a new product engineering methodology. A research hypothesis is formulated on the new methodology. The methodology is applied in the industrial setting and the results of these experiments are observed and used to evaluate the hypothesis. Coupled to the multi-disciplinary design problems for high-tech systems discussed in the beginning of this chapter, the research hypothesis of the Boderc project was chosen as:

> *The product creation lead time will be reduced significantly by the use of multi-disciplinary models during the early product development phases.*

The term *Carrying Industrial Partner (CIP)* is used for the company that provides the problem and the industrial setting. The CIP of Boderc is Océ Technologies, which creates high-volume document printing systems.

The industrial context

One of the product families that is designed by Océ technologies is a range of high-volume printers and copiers, see Figure 1.5.

Figure 1.5: The Domain: Printers and copiers by Océ

The application context is best characterized by document printing systems that are highly productive, reliable, and user-friendly. These systems can print on several sizes of media, different weights, automatically on both sides and include stapling, booklet production, or other types of finishing. In order to be perceived as reliable devices, such copiers must be very robust with respect to variations in media. As the printing speed is rather high (typically above 1 image per second), timing requirements are tight and advanced mechatronics are indispensable. This indicates that variations in timing parameters that relate to paper and image transport must be controlled up to a high degree. This becomes the more apparent if one realizes that the positioning of images on paper has tolerances well below 1 mm.

When considering the embedded control of these systems, one should think of controlling multiple sheets that travel the paper path simultaneously and synchronizing this sheet flow with the imaging process. In Figure 1.6 overviews of a copier are presented. When the copier is in normal operation, a sheet is separated from the trays in the paper input module (PIM), after which it is sent to the paper path that transports the sheets accurately in the direction of the print engine, where the image is fused on a sheet of paper. After that, the sheet is turned for duplex printing, or transported by the paper path to the finisher.

1.2 Multi-disciplinary methods

The Boderc research falls typically within the category of multi-disciplinary design methods as opposed to the more conventional mono-disciplinary research areas like mechanical, electrical or software engineering. The latter research fields are relatively mature, although some doubts exist about the maturity of software engineering [91]. Some bi-disciplinary approaches exist, for instance hybrid systems theory [103] that

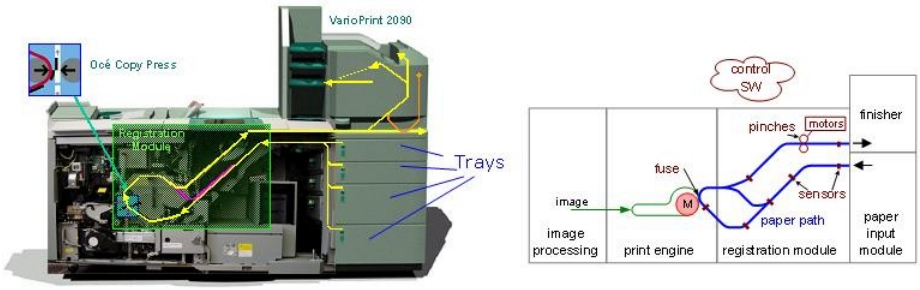


Figure 1.6: Illustration and schematic picture of a copier

combine continuous dynamical models (using e.g. differential equations) typically describing the physical part of a high-tech machine and discrete models (e.g. finite state machines or automata) to described the software behavior. The hybrid field is relatively immature and many issues are at present unsolved (at least at the large-scale needed for industrial usefulness). However, the industrial need for analysis / synthesis methods for high-tech machines in which this 'hybrid interaction' plays an important role, will stimulate the research in this domain over the years to come.

Researchers in the mono-disciplinary areas are used to well-defined problems that can be studied in depth with solutions most often based on mathematical rigor. A lot of uncertainty pops up when we move to multi-disciplinary problem solving. The problem itself is only partially defined, while at the solution side different formalisms have to inter-operate, such as discrete (software) and continuous (mechanical) models. Figure 1.7 shows a categorization of the design methods with as vertical axis the degree of multi-disciplinary interaction. The form of the method is an indication how well the method is defined and how much uncertainty is left.

In the industrial context the *system* level is often relatively well-defined in a systems requirement specification. Such a specification describes the functionality of the system and quantifies the main performance characteristics. The translation of these requirements into mono-disciplinary design choices, however, is still full of uncertainty. A lot of uncertainty is caused by the many (dependent and interfering) design dimensions that have to be managed at the same time. In Figure 1.7 the methods at this level are called *multi-objective design methods*.

The translation of system requirements to detailed mono-disciplinary design decisions spans many orders of magnitude. The few statements of performance, cost and size in the system requirements specification ultimately result in millions of details in the technical product description: million(s) of lines of code, connections, and parts. Figure 1.8 shows this dynamic range as a pyramid with the system at the top and the millions of technical details at the bottom.

The methodologies to be established by ESI, including the Boderc results, address the multi-disciplinary area and aim at coupling the academic research to industrial

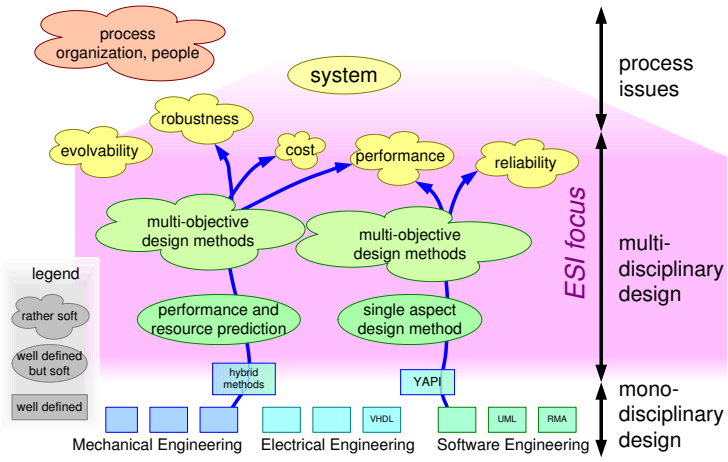


Figure 1.7: From mono-disciplinary to system design

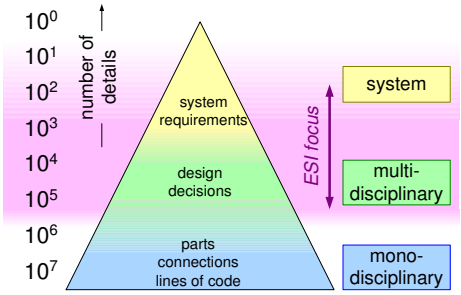


Figure 1.8: Exponential pyramid

practice. In Figure 1.7 this is the range from *single aspect* to *multi-objective* design methods. In the pyramid, Figure 1.8, it is the area of translating hundreds of system level requirements into tens of thousands of design choices.

The Embedded Systems Institute

Boderc is the first project in a long line of ESI projects within the field of multi-disciplinary creation methods. This is a young research field, which is called *embedded systems engineering* (ESE) by ESI. The existing scientific disciplines have little experience in this field, most experience can be found in industry.

The mission of ESI is *to advance industrial innovation and academic excellence in embedded systems engineering (ESE)* with its vision *to create and apply together with*

its partners world-class ESE methods. The developed methodologies must support all aspects of the creation: specification, design, integration, test and validation.

1.3 Reading guidelines

This book contains a selection of the main outcomes of the Boderc project. In the Figure 1.9 an overview of the book is presented that can be used as a reading guide. The introduction (Chapter 1) has almost ended. After the introduction we present in Chapter 2 the Boderc design methodology, which consists of a system-level reasoning framework and plug-ins (modeling formalisms and techniques) that are used during the reasoning to get more in-depth insight. The Boderc methodology consists of a method part, which forms together with sub-methods like the key driver method (Chapter 3), threads of reasoning (Chapter 4) and budget-based design (Chapter 5) the reasoning backbone of the Boderc methodology.

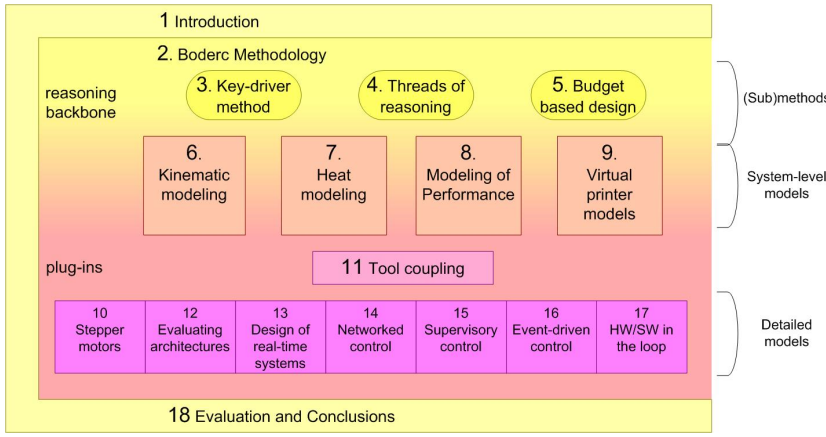


Figure 1.9: Overview of the contents of the book

This reasoning backbone is mainly qualitative. Once important design choices are identified and the tension and conflict in the choice are known, often quantitative information is needed to make a well-founded tradeoff. Chapter 2 discusses this process in detail. To obtain the quantitative information one can retrieve this from previous projects, figures of merit, rules of thumb, data sheets, et cetera. If this information and resulting insight are not sufficient to make a proper design tradeoff, an in-depth model-based study is often used. This is where the plug-ins come into play.

Chapter 6 describes one of the most successful models of the Boderc project being the Happy Flow model that studies the design of the mechanical lay-out of the paper transport system and the scheduling of print jobs. This chapter presents the main characteristics of the Happy Flow model and identifies the main industrial success factors.

These factors can be used as a stepping stone towards guidelines on how to create models that are industrially successful. Important aspects of a copier are the heat and power flows, especially since environmental constraints are becoming tighter and tighter. In Chapter 7 a modeling approach is presented that studies these issues. How to evaluate the overall control architecture in terms of response times, CPU load, et cetera, is discussed in Chapter 8. Chapter 9 describes models that are related to printing quality. New printer technologies are assessed via 'virtual' printer models on their printing quality. The models described in Chapter 6-9 are positioned higher in Figure 1.9, when compared to Chapters 12-17. The reason is that these models have a more 'system-level character' as they describe system aspects or large subsystems of the copier. Roughly speaking, the models presented in Chapters 10-17 have a more mono-disciplinary and detailed nature.

Océ Technologies traditionally used DC motors as drives in the paper transport system. However, there were important reasons to replace the DC motors by stepper motors. Chapter 10 investigates the feasibility of stepper motors for this purpose and aims at building an understanding of stepper motors that lead to practical design rules. Chapter 12, gives an overview of several state-of-the-art performance analysis for embedded system architectures for real-time systems. A case study inspired from industrial practice will be used to compare the performance analysis techniques. Based on these experiences, an indication will be given which method is best used under which circumstances to successfully support the decision making process for the architecture. Chapter 13 presents a model-driven design approach for real-time systems. This approach enables the analysis of real-time systems and allows automatic software code generation from the model that preserves the properties analyzed in the model. Chapter 14 has a more control engineering view as it analyses the effects of jitter and latencies (communication and computation delays present in any real-time system) on the control (servo) loops present in copiers. Latencies are inevitable and their effects can be disadvantageous with respect to stability and control performance. This chapter gives analysis methods and techniques for the synthesis of controllers that are robust against jitter and latencies.

For the control design of the drives of the paper transport system (based on the schedules as computed via the Happy Flow model of Chapter 6) in Chapter 15 a hierarchical control paradigm based on supervisory control is proposed. A systematic analysis and design procedure based on low-level controllers for the motors in combination with high-level sheet control is proposed and verified via both simulations and experiments.

Chapter 16 describes the design and application of event-driven control. Event-driven control abandons one of the severe requirements that are often posed by control engineers on the real-time implementation of their algorithms. The conventional fixed sample time is removed and novel control algorithms are described, which allow for control updates being triggered by events (e.g. the arrival of new measurement data), rather than by progression of time. Event-driven control can have major benefits with respect to resource utilization like processor and communication load, while still main-

taining a good control performance. A particular event-driven controller was experimentally validated for the image control in a printer prototype with good control and software performance.

In the next chapter, Chapter 17, a systematic design trajectory is proposed for the combination of real-time controllers and the physical / mechanical process. On several levels the interactions between the hardware (processing platforms and implementation) and software aspects of controller-plant interaction are studied. A design path is indicated in which stepwise the original (simulation) models of both plant and controller are replaced by their real implementations. Chapter 11 is related to Chapter 17 as it discusses ways to simulate real-time embedded software together with its environment being of a physical / mechanical nature. Via the coupling of tools from software engineering (that model the control software of the system) and simulation tools from the mechanical/physical domain (modeling the physical part of the system) one can inspect if the software-plant combination is functioning properly. In this sense Chapter 11 is positioned closer to the system level than the detailed models in Chapters 15-17.

In the last chapter we will evaluate the overall project results and discuss its impact, spin-off and lessons learned.

Chapter 2

A design methodology for high-tech systems

Authors: W.P.M.H. Heemels, E.H. van de Waal and G.J. Muller

This chapter is a re-worked version of a paper for the Conference on System Engineering Research (CSER) 2006 by the same authors and with the same title.

2.1 Introduction

As already mentioned in Chapter 1, there is a need for a framework that supports efficient evaluation of design choices over multiple disciplines. Actually, evaluation of design choices over multiple disciplines is one of the important features of the emerging field of Embedded Systems Engineering (ESE). Typically, ESE for high-tech machines is performed by highly experienced individuals, using mostly intuition and 'gut feeling'. The experience of these individuals is hard to transfer, thereby limiting the speed with which companies can develop new products. This way of working is effective when the project remains small and limited to one location, where a relatively small number of people are involved in the design. However, to enable the co-operation for larger projects across multiple sites, an ESE framework is needed. Even for smaller projects, such a framework is expected to speed up the design process and to reduce integration time. Moreover, an ESE framework that captures the way of working of the experienced architects should enable junior architects to learn the skill of system engineering faster. Hence, in this respect the formulation of the design methodology has both an educational as an industrial application character.

In System Architecting (SA) research, some frameworks have been established (see, e.g. [89], [77] or Chapter 4 in [81] for an overview). Also, academic research has produced techniques that could be useful in industry. However, these find very

limited use, see e.g. [94] and [83], and the need for multi-disciplinary methodologies is still large as expressed in [82]. In [82] several reasons are mentioned that hamper the creation of such methodologies. Lack of description and lack of connection of the higher level design methodology to mono-disciplinary methods are just two. The lack of connection hampers the use of frameworks and methodologies for (very) large scale systems (e.g. aerospace and military) in the *technical* development and realization of high tech systems like a copier. This also reflects the slight difference between SA and ESE as ESE particularly focusses on the connection between the system level and mono-disciplinary methods. By lack of description we mean that although multi-disciplinary methods exist and are in use in the industry in various domains, their use is very implicit - typically 'gut-feeling'-based as mentioned before. The consolidation of these industrial methods is very poor. The lack of explicit description means that a lot of open issues remain. Open issues erode the value of these multi-domain methods. To tackle the lack of description and connection to mono-disciplinary techniques, this chapter presents an attempt to explicitly describe such a multi-disciplinary methodology and give place to mono-disciplinary design techniques. As [77] states, at high levels of complexity, analytical methods are no longer sufficient and heuristics come into play. In this design methodology, heuristics and analytic rigor find their place in the high-level method and the mono-disciplinary techniques, respectively. The usefulness of the proposed methodology here is largely due to the connection between the two. Moreover, by making the methodology explicit, discussions should be triggered on the open issues that require future research.

This chapter proposes the Boderc *model-based* design methodology that consists of formalisms, techniques, methods and tools:

Formalisms are languages / syntax used for system modeling. Formalisms exist for modeling behavior, but also to formalize system requirements. Instances of formalisms are called *Models*. Examples of formalisms are differential equations, (timed and hybrid) automata, finite state machines, temporal logic and queuing formalisms.

Techniques are used to retrieve information from models or to transform models. Examples of analysis techniques are model checking, performance analysis and program analysis techniques. Examples of transformation techniques are high-level synthesis and software compilation.

Methods ('reasoning frameworks') provide guidelines and can be seen as a 'recipe book' how and in which order to apply certain *Formalisms*, *Techniques*, *Submethods* and *Tools* to solve the design problem at hand. *Methods* are ways to 'capture' design and reasoning knowledge of experienced modelers and designers. A method indicates for instance decomposition in steps (possibly techniques) and an order in which the step should be performed.

Tools: Software *Tools* support the efficient application of *formalism*, *techniques* and *submethods*.

2.2 Boderc design methodology

When developing high-tech machines as described in the introduction, industrial constraints like project duration and available man power are paramount and as such they were explicitly stated in the Boderc goal, see Figure 1.2. These constraints must be deeply integrated in any successful methodology. To meet these constraints, a careful selection has to be made on how to invest design effort and time. The methodology provides two means:

- Focus the in-depth analysis (via modeling) on the most critical issues, preventing 'wasting' effort on less relevant problems. For this, one has to identify the *most essential* conflicts and tensions from the design decisions to be made.
- Using simple models that create insight in a design decision within a reasonable time (hours, weeks), instead of detailed models that requires months or even years to develop. The right level of detail must be chosen, which can range from back-of-the-envelope calculations to very detailed models depending on the accuracy of the answer needed. Stepwise refinement of models, typically starting with back-of-the-envelope and then extending towards more detailed models, can be useful for this.

Even when using models, physical prototypes are essential because of the confrontation with physical reality, where overlooked issues will inevitably pop up. However, it is difficult to quickly evaluate different designs through physical prototypes because a new prototype is needed for each design. Through analysis of models different designs can be evaluated much faster. As a consequence, both models and prototypes are indispensable.

Another benefit of the methodology is that it gives place to formalisms and techniques (which can be seen as 'plug-ins' in the method) and when they should be applied. Documenting the conditions under which academic formalisms/techniques and industrial state-of-the-practice are applicable and effective and their level of prediction accuracy form valuable information. Moreover, gaps can also be identified that require future research (e.g. extending state-of-practice and 'industrializing' state-of-the-art academic techniques) to obtain the right abstraction level for industrial practice.

2.3 Linear stepwise version of the method

The high-level 'method'-part of the methodology is given as the collection of the following steps:

1. **Preparation of the design**
   (a) Identify (customer) key drivers and requirements
   (b) Identify realization aspects of concern

(c) Make core domain knowledge explicit

2. **Selection of critical design aspects**

   (a) Identify tensions and conflicts (qualitative)

   (b) Gather facts and identify uncertainties to quantify tensions and conflicts

3. **Evaluation of design aspects**

   (a) Build small models (small = hours to weeks of effort)

   (b) Perform measurements

Note that there can be other (sub)methods that are a part of the design methodology that support the design of subsystems, e.g. control engineering has its own methods to synthesize control algorithms. These submethods can be used when the control system of the high-tech machine has to be designed.

The above steps in the (high-level) method are to be used iteratively, so that progressive knowledge can be used. In Figure 2.7 the iterative nature and the dynamic flow of information between the steps is reflected better. Below, the steps and corresponding submethods, techniques and formalisms will be explained in more detail. Good visualization of the outcomes of the steps is important to create insight and overview. The design of a high-volume copier will serve as a means to illustrate the individual steps.

2.4 Step 1: Preparation of the design

In step 1, a good understanding of the product to be developed has to be achieved and existing knowledge is gathered to be available for the new design.

2.4.1 Step 1a: Identify (customer) key drivers and requirements

In step 1a, the goal is to identify why a customer (or other stake holders like the internal business strategist) would want the new product. The main drivers for the stake holders should be identified and insightfully related to system requirements. This is linked to the product business case. This can be achieved using activities like interviewing marketing experts, interviews and workshops with customers, story telling [81], et cetera. The results of these activities can then be summarized using a high-level requirements engineering technique. The *key-driver model* has been found to be very useful for this purpose (see Chapter 3).

Example: As part of the Boderc project, the submethod of key driver analysis (see Chapter 3) was applied to a high-volume copier. The key drivers of the copier were identified and refined into application drivers and finally into system requirements. This analysis is explained further in Chapter 3. Already a part of the key driver model is shown in Figure 2.1 below. The complete key driver model can be found in Figure 3.6.

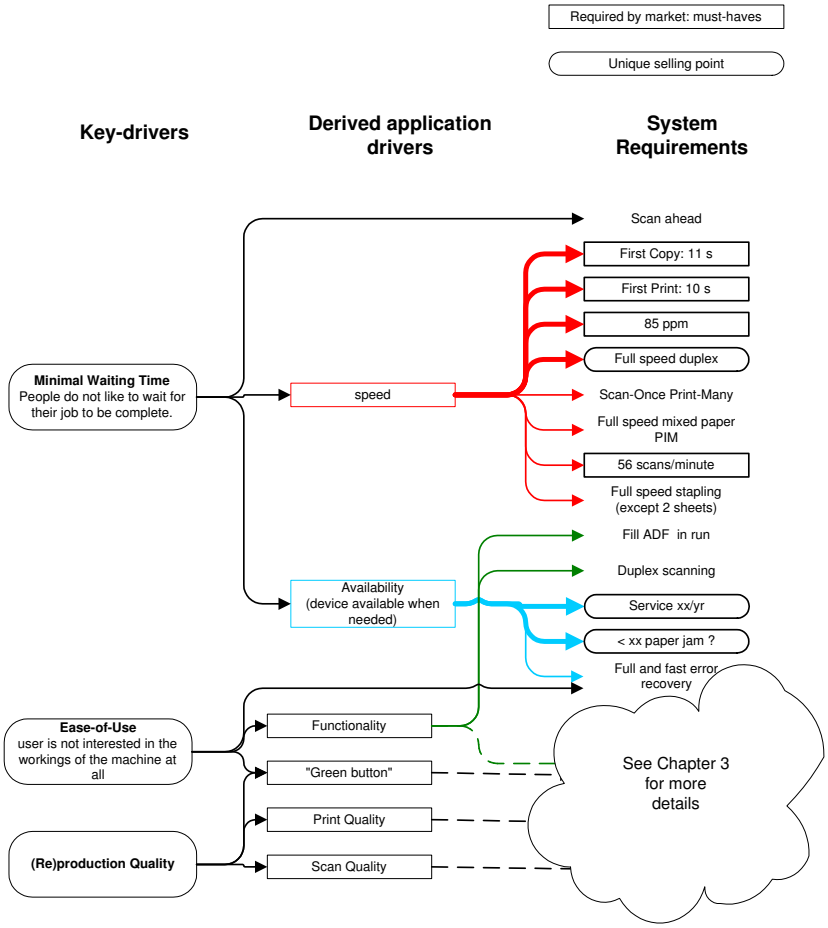


Figure 2.1: Part of a key driver model for a copier.

2.4.2 Step 1b: Identify realization aspects of concern

In step 1b, the goal is to identify which designs aspects of the machine are of concern for its marketing success. In practice, typical the issues or worries that are hot during (coffee or lunch) discussions between the design engineers form good starting points. Some of them might be non- issues caused by non-rational fears, uncertainty, rumors, et cetera, others might be critical and jeopardizing the success of the product. This has to be found out in steps 2 and 3. In step 1b they are only identified. Currently, there are not many concrete (sub)methods and techniques that can be used for this activity; thus this is an interesting area for further research. Methods like ‘story telling’ [81] and scenario or use case based reasoning (see e.g. [23]) can be used for this activity.

Checklists with problematic issues in previous projects (typically input coming from step 1c) can be used in step 1b. The introduction of new technology, new (more strict) environmental regulations and successful or failing competitors (and in particular the reasons of success of failure) should always be considered with caution.

Example: Based on experience of previous projects and the more stringent power norms nowadays, maximum power usage was an issue in the design of the copier. Also the introduction of stepper motors in the copier is a worry as traditionally DC motors were used.

2.4.3 Step 1c: Make core domain knowledge explicit

In step 1c, the goal is to make the most important lessons that were learned during the design of previous machines explicit. In most companies, this knowledge is only known implicitly: it is stored in the minds of key designers. By making this knowledge explicit, a common understanding can be achieved amongst engineers. Capturing the *context* in which a certain design was successful, can be useful to solve similar problems in a same manner in a new machine without much efforts (in Figure 2.7 indicated by 'no-brainers'). It prevents re-inventing the wheel. Going outside the context with a particular solution should be done with caution, which could require to consider it as an 'aspect of concern' and thus part of step 1b. Context is an important factor in design success.

The goals of this step can be achieved by investigating the models, design solutions, methodologies, and so on, used in previous designs. Especially designs that were not successful are useful to investigate (see also 1b above). The main question is why things were done in a certain way. The results of this investigation must then be summarized, e.g. by identifying design patterns, by writing tutorials and white-papers, determining rules-of-thumb, et cetera. Of course, part of this information is hopefully consolidated at the end of previous projects, so that this is readily available. Industrial practice often turns out otherwise.

Example: Figure 2.2 shows some core technologies for designing copiers: the main system architecture, the paper-time diagram used for analysis of the timing of print jobs in the paper path, and the main components used in the paper path.

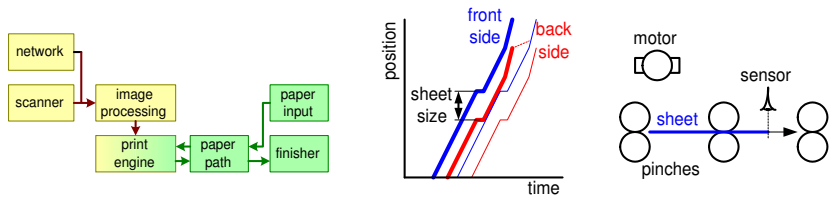


Figure 2.2: Examples of core domain knowledge for a copier manufacturer.

This information is very well known for most experienced copier designers, and

these items are always used. However, the familiarity has subtle dangers in that people forget the reason why these technologies are used, what the limitations and advantages are, and when alternatives should be used. Thus there is benefit in formally 'documenting' this knowledge. Of course, this has to be done in an effective and structured manner. Models can also be used to capture this knowledge. A good example is the Happy Flow model (see Chapter 6), which is used for the design of the layout of the paper path and the scheduling of print jobs. One of the reasons for industrial success is that it contains a lot of core domain knowledge in an easily accessible model.

2.5 Step 2: Selection of critical design aspects

When designing a new product, issues arise constantly. It is imperative to differentiate between important issues, which imply a great risk to the project if not dealt with adequately, and non-issues. Otherwise much time is lost over unimportant issues making development prohibitively expensive. In step 2 the design aspects of concern found in 1b are prioritized by their importance or value for a customer (as analyzed in step 1a), by how hard it is to solve the problem, and how sensitive or vulnerable the overall system is to this challenge.

2.5.1 Step 2a: Identify tensions and conflicts (qualitative)

In step 2a, the goal is to identify *qualitatively* the design tradeoffs and essential tensions that are coupled to a certain aspect of concern (1b). The fact that a design issue is of concern implies that it must have both benefits and drawbacks (in terms of key drivers and system requirements found in step 1a). Making the tensions and conflicts between benefits and drawbacks explicit allows them to be treated systematically throughout the design process.

A good submethod to find these tensions and conflicts is *threads of reasoning*, which investigates where the real tradeoffs are in a design. Concrete design choices are linked to key-drivers and negative side-effects pop-up. In Chapter 4 the submethod is described and applied for the digital control architecture in a copier. The threads of reasoning diagram for the case study is presented in Figure 2.3. Also the 'question generator' [81, Section 9.2.3] supports the exploration of design tensions. Organizing workshops and brainstorm sessions is another means. In a workshop, several experts from different disciplines are invited to work together on the main concepts of the machine. They will very quickly find the tensions and conflicts in the design by bringing their own concerns and worries across and connecting them.

Figure 2.3 is at the heart of the Boderc methodology. For several design choices (e.g. the use of stepper motors instead of DC motors) the relations to the drivers for the copier are displayed. The main benefit for the use of steppers is its low cost price. However, a drawback is the limited positioning accuracy and thus possible problems for the printing accuracy, which is a customer application driver (see Figure 2.1). The

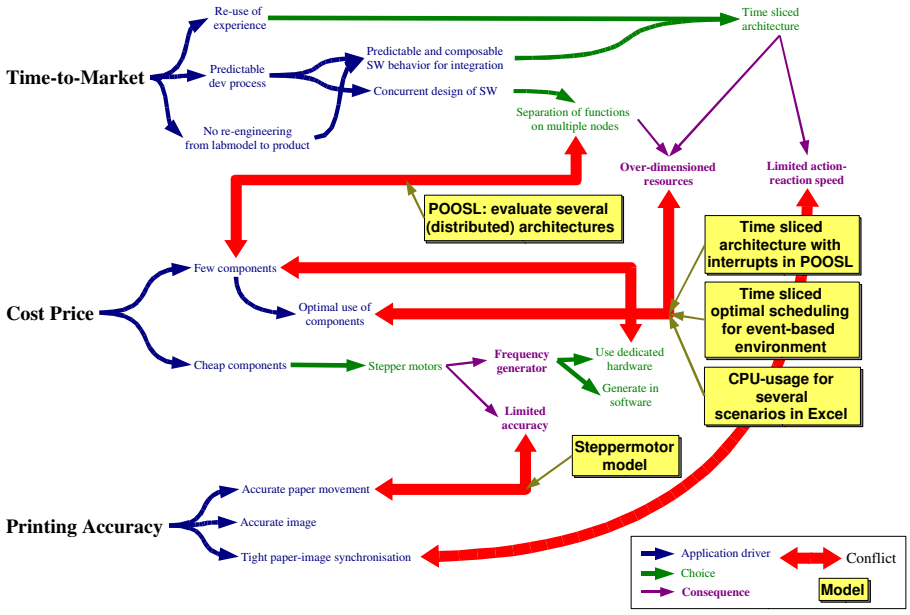


Figure 2.3: Visualization of threads of reasoning for the control architecture of a copier.

conflict between cost price and printing accuracy and several other conflicts are indicated in the figure (see legend for color use). These require further investigation in step 2b. If the results from 2b are inconclusive, an in-depth study is required according to the steps 3a and 3b. The rectangles indicate the models that have been used to create more insight in the conflicts.

2.5.2 Step 2b: Gather facts and identify uncertainties to quantify tensions and conflicts

In step 2a, tensions and conflicts were identified qualitatively. In step 2b, the goal is to select those tensions and conflicts that require further study. Often, the tensions and conflicts are a result of worries, uncertainty, lack of facts, non-rational fears and turn out to be non-issues. These can often be unmasked by quantifying the issues with rough estimates, using simple facts to weed out the fears, thereby diminishing the worries in the organization and enabling it to focus on the important issues. However, there will be some issues where there is insufficient knowledge to make an intelligent decision. In those cases, further study is warranted (step 3).

There are many ways to find the facts needed besides using the core domain knowledge of step 1c. For instance, an expert can be asked (use the question generator mentioned above), or rough orders of magnitude can be estimated. Also, figures of

merit from previous designs can be used. Finally, much knowledge is readily available through existing literature. Facts from all these sources can be used to discard irrelevant conflicts. Note that making quantitative assumptions, which all engineers have in their mind, explicit will also reveal the (qualitative) tension. So, step 2b often also precedes step 2a in practice.

Risk assessment (see e.g. Chapter 6 in [89]) is one way to select the tensions that should be addressed more thoroughly as they consider both the impact and the probability of occurrence of a particular issue. Also back-of-the-envelope calculations can be a good starting point as they do not require much effort and time and give first estimates. Iterative refinement to more complicated models is a good means to progressively analyze a tension. Determining a budget (see Chapter 5 for details) which distributes a resource over different parts of the machine is a formalism that is often used in practice to determine the real magnitude of a problem which is too complex to analyse at the top level. Of course, there is no strict boundary between the current steps 2b and 3a: it is not always clear when to categorize a back-of-the-envelope calculation in step 2 and when to associate a model to step 3a. But in order to keep track of issues, e.g. to allow proper project management, it is helpful to make an *explicit* decision to further study an issue by placing it in step 3a.

Example: In the Boderc project, it was investigated if it were better to use stepper motors or DC motors in a specific copier configuration. For an initial investigation, the thread of reasoning was extended with numerical data on cost price, life time, et cetera. To assess the consequences of the implementation of steppers for important machine characteristics, a risk assessment matrix model (see [89], Chapter 6) was created (Figure 2.4). From this matrix, issues that required in-depth investigation were identified.

	Uncertainties	Impact	Result
Cost price	**1**	**10**	**10**
Lifetime	**3**	**3**	**9**
Accuracy (reliable)	**10**	**9**	**90**
Ease of design (time)	**7**	**5**	**35**
Noise	**5**	**6**	**30**
Efficiency (power)	**3**	**3**	**9**

Figure 2.4: Quantified threads of reasoning for the use of stepper motors in a copier.

2.6 Step 3: Evaluation of design aspects

If the facts in step 2 are not sufficient to make a decision, the issue needs to be evaluated properly. There are two ways to do this: either using a model-based approach or measurements on prototypes. Of course, it can also be a mixture of the two, e.g. to validate models. A final validation of a product is of course always the final integration before production.

2.6.1 Step 3a: Build small models

In step 3a, the goal of this stage is to resolve an open conflict found in step 2 using simple (small) models. As mentioned, models are often very efficient in evaluating design options, as models can be readily modified whereas prototypes are harder to modify. Also models might create a deeper understanding of the relationships in the tension.

A key issue when using models is which formalism to use to answer the question at hand. Often, model formalisms are suggested in the core domain knowledge gathered in step 1c . If this is not the case, some literature study or research may be required to find the right formalism.

A second key issue is to find the right abstraction level and model boundary to answer the question with the right certainty. The goal is to keep the modeling effort as small as possible.

An interesting question is whether the model is based on theoretical (physical) knowledge (sometimes called first principle or white box modeling), or on empirical facts (regression, identification or black box modeling). Depending on the case at hand, one might prefer one over the other.

Example: Below are some examples of models used in the Boderc project. Already in Figure 2.3 some models have been mentioned that were used to analyze specific tensions further. Other modeling formalisms and techniques that have been used include:

- Performance analysis techniques to predict and evaluate the real-time behavior of the copier control software running on hardware platforms (see e.g. Chapters 8, 12 and 13).
- Evaluation of real-time embedded control software via model-based simulation of its environment is discussed in Chapter 11.
- The Happy Flow model, which supports the design of the paper transport systems in the copier. Strong visualization and animation complement the models. These are based on ‘good weather’ conditions: lower level (dynamical) phenomena of motors, slip, jitter and delays in control loops, et cetera, are not included. See Chapter 6 for an explanation of this model.
- Dynamical models including software execution times of part of the paper path around the fuse, where paper and image meet and accurate synchronization is needed [25].
- Power budgets as visualized in Figure 2.5 are also used to understand power flows through a copier. Thickness of arrows in the figure is related to the amount of power flow. See Chapter 5 for more details. Note that a budget itself is a model with an underlying modeling formalism. In Chapter 5 a (sub)method is even described on how to create budgets.
- Virtual printer models to assess the printing quality of new printer technologies are presented in Chapter 9

- Dynamical models and techniques that analyse and predict the behavior of stepper motors (power usage, vibrations and resonances, positioning accuracy, et cetera). In Chapter 10 an overview of the modeling activities is given.

- Event-driven control models (Chapter 16) are derived to study the control performance and the processor load of a new type of controllers that are used to control the motion of the image.

- And many others as documented in this book.

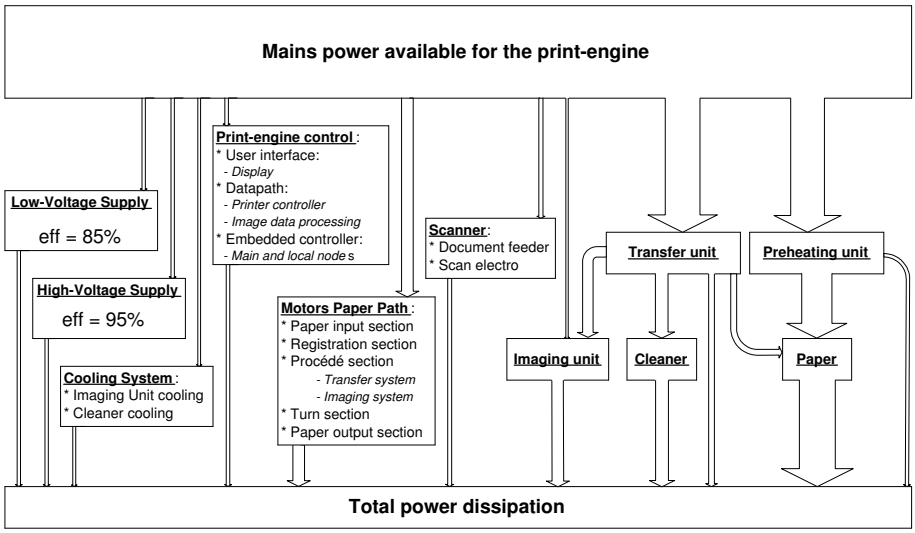


Figure 2.5: Visualization of the power flow through a copier.

2.6.2 Step 3b: Perform measurements

In step 3b - just as 3a - the goal is to gather the facts with which issues from step 2 can be dismissed by using measurements. Measurements can play two roles. Either they are used to tune the models to describe practice closely (parameter estimation, identification, model validation) or try to resolve a conflict directly - without a model - by using dedicated experiments. As measurements are from the 'real world', they are usually more authoritative than results from models. However, not every phenomenon can be measured readily, for example because sensors can not be inserted (the place where you would like to measure cannot be reached) or sensors disturb the phenomenon. It is difficult to determine the effects of parameter variation from measurements. Also, measurements can be faulty. Thus sanity checks are always required.

Example: A model was made of the dynamic behavior of the motors in the paper path [18], as mentioned before. This model was validated with measurements from a real motor in the copier being modeled (see Figure 2.6). Within the Boderc project, also measurements have been performed on hardware platforms to evaluate their real-timing behavior (e.g. the influence of caching in micro processors), see Chapter 8.

Often, it is very beneficial to have short iterative loops where measurements and modeling activities follow each other. The measurements show where the models can be improved and the models explain the measurements and show how design choices would influence the results. Models can often capture the relationships between system properties better than a finite number of measurements. Towards the end of a development project more and more the emphasis will shift from modeling (step 3a) towards prototyping and building the actual system (step 3b).

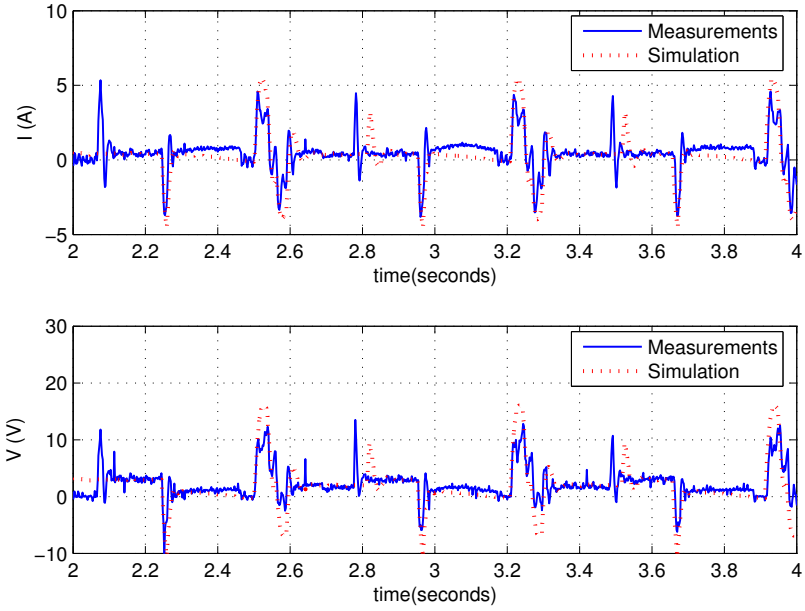


Figure 2.6: Simulation versus measurements for a single motor in the paper path.

2.7 The method as a structured chart

The nice step-plan shown in the previous section is iterative as depicted in Figure 2.7. This figure contains the same steps, but shows the dynamic flow of information and the making of decisions. For instance, once step 3 has given conclusive answers on a particular issue of concern coming from 1b via step 2, a design decision can be taken.

The iteration now proceeds to a next issue of concern. However, also important information obtained during the in-depth study of the previous issues (e.g. data, models, design patterns) should be consolidated in the core domain knowledge (step 1c).

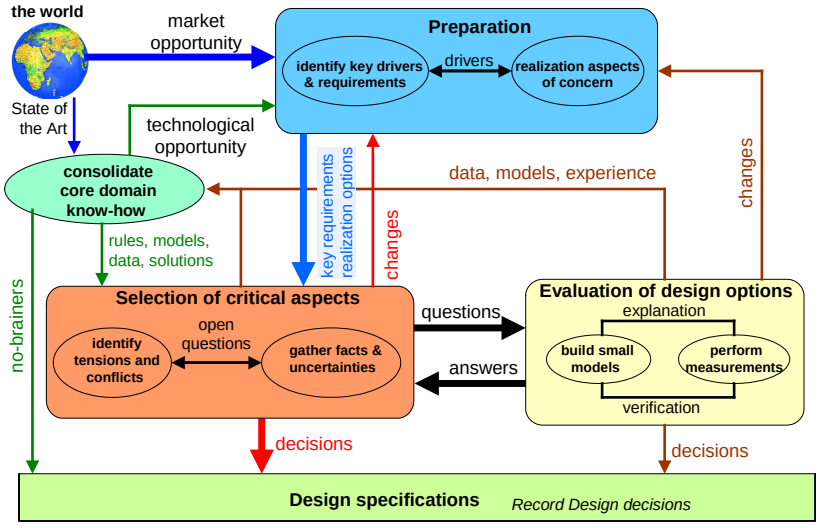


Figure 2.7: Dynamic flow of information in the method.

2.7.1 A ‘schoolbook’ example of Boderc reasoning

To give a good example on how the Boderc methodology and reasoning works we give a quick preview of the selection of the parameters of an event-driven control algorithm to control the position of sheets in the copier. Details on the particular case can be found in Chapter 16 on event-driven control. This particular problem might at some point be identified in step 1b as a realization aspect of concern. In step 2a the tension in this design issue can be found using threads of reasoning. It turned out that a trade-off had to made between control performance (e.g. tracking errors and disturbance attenuation), which is related to the key driver printing quality on one hand, and software performance (processor load of its implementation), which is related to the cost price (as a smaller CPU or fewer CPUs can be used) on the other. After collecting some figures-of-merit in step 2b it was concluded that more quantitative information

would be necessary. A detailed dynamical model (using Matlab/Simulink) was made that could predict the control performance (in terms of maximal tracking error $e_{\max}$) given the controller. By varying the main control parameter (e_T) we could express the control performance as a function of this parameter. Using the outcomes of this simulation model, also the number of control computations could be derived. Together with micro-measurements (step 3b) on the platform the controller would be implemented on, a prediction could be made of the processor load (in terms of total computation time over a given time interval). Both graphs are given in Figure 2.8.

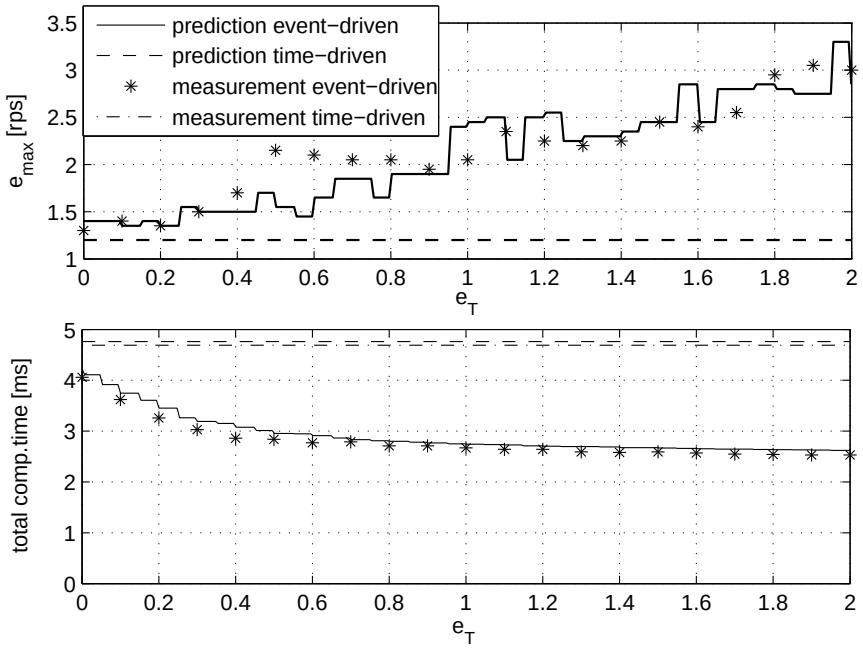


Figure 2.8: Both control performance and software performance as function of the main control parameter e_T.

These two graphs are returned to phase 2b in which this is the necessary quantitative information to make a well-founded trade-off by a system architect as he can now oversee the consequence in both domains. Actually, the predictions in Figure 2.8 were validated later by measurements on a prototype.

Typically, in the way of working above, in-depth detailed models were used to make predictions. Not the whole model was given to step 2, but only system-level abstractions of it. Of course, both the models as well as the curves leading to the decision can now be consolidated and documented as core domain knowledge (step 1c) and another realization aspect of concern can be considered next. Sometimes the in-

depth study might also trigger new questions and new aspects of concern. For instance, maybe there is not a satisfactory value for the control design parameter that satisfies the requirements. This might trigger a reconsideration of the processing platform on which the controller is implemented. As only micro-measurements (or benchmark numbers) are needed to perform the prediction of the processor load, the modeling effort can also support the selection of the processing platform.

2.7.2 Placing the Boderc activities in the Boderc methodology

Figure 1.9 in the Introduction has already given a rough positioning of the Boderc activities within the methodology. Typically, the modeling formalisms and corresponding techniques for studying particular aspects or subsystems of the to-be-developed machine can be seen as 'plug-ins' that are used in the steps of the method. For several aspects (e.g. printing quality, power usage, throughput, et cetera) and different parts of the copier (e.g. ranging from detailed models for particular stepper motors and parts of the digital control architecture to system level models of the complete paper track), models and corresponding techniques were are applied. In view of the design pyramid in Figure 1.8 these models have various levels of detail. Typically, Chapters 6-9 are more system level models and Chapter 10-16 describe the more detailed models. Both categories can be used as 'plug-ins' in step 3a to get quantitative information and insight in the qualitative tensions identified in step 2a. Next to the overall method of the Boderc design methodology given in Section 2.3 there are three submethods: Key driver method (Chapter 3), Threads of reasoning (Chapter 4) and Budget-based design (Chapter 5). However, when designing subsystems one might use mono-disciplinary submethods. For instance, for the synthesis of control algorithms one might use typical design methods as available within control engineering.

2.8 Conclusions

As we are all aware, there is a strong need for multi-disciplinary methodologies that support the system architecting process. In [82] various reasons are mentioned that hamper the creation of such methodologies. In this chapter we presented research to overcome two of them: lack of description and lack of connection to mono-disciplinary techniques. This resulted in an emerging design methodology that was stated in an explicit manner. The methodology consists of a reasoning framework in the form of a multi-step method, modeling formalisms, analysis techniques and tools. By giving place to modeling and analysis activities, which can be mono-disciplinary, a first step is made in connecting the multi-disciplinary method to mono-disciplinary techniques.

Previous to writing this chapter, some steps were taken to validate the methodology. Although various issues remain open, we can already draw the following conclusions:

- The Boderc methodology mimics the way of working of a senior system architect. For instance in [70] the steps of the method can be recognized in the

evaluation of an architecture for a DVD hard-disk recorder. As shown by [70], applying the steps of the methodology can prevent system architects from falling prey to ill-founded non-quantitative reasoning, which can lead to trade-offs based on incorrect assumptions instead of on quantitative arguments and facts.

- Discussions with junior and senior system architects from Philips revealed that there is a clear recognition of the steps in the method. It matches their way of working and it makes that more explicit. They acknowledged the value of the methodology. Of particular interest for them were the visualizations, e.g. the key driver model in Figure 2.1 and tensions and conflicts in a thread of reasoning diagram (Figure 2.3). Documenting design decisions and capturing the main arguments in insightful overviews were considered particularly valuable. Also various models as, for instance, the Happy Flow model (Chapter 6) contain a lot of implicit design decision, which can be extracted.

- The application of individual modeling activities (using formalisms and techniques) on particular industrial problems (e.g. paper flow scheduling, stepper motor dynamics analysis, et cetera) were considered beneficial by Océ.

Of course, many issues are still open within this methodology, as in the whole field of multi-disciplinary design. For instance, finding the right level of abstraction for modeling formalisms in an industrial setting is hard. Many academic (mono-disciplinary) formalisms are too complex and many state-of-practice formalisms are too coarse. Finding the right balance between them is an important issue for future research. Extending the design methodology by further formalisms and tools (especially selecting design aspects of concern in step 1b and selecting critical design issues in step 2b) is also open. Hence, by making an attempt to be explicit, this chapter hopefully initiates many discussions, allows further validation of the design methodology and stimulates future ESE research.

Chapter 3

The key driver method

Authors: W.P.M.H. Heemels, L. Somers, P.F.A. van den Bosch, Z. Yuan, B. van der Wijst, A. van den Brand and G.J. Muller

This chapter is a re-worked version of a paper for the International Conference on System Engineering and Applications (ICSSEA) 2006.

3.1 Introduction

The complexity of the products being designed by industry today is increasing at an astonishing rate. To keep the design of complex machines focused, it is important to know the essential customer objectives and to relate these to those system requirements that have the largest influence. A structured graph showing these relations helps to keep an overview of the overall design. In particular, specifications can be easily traced back to the objectives of the customer. The *key driver method* is one of the possible means to obtain the system requirements in a systematic way and to provide such a structured overview. The key driver method fits in the broader framework of the CAFCR methodology [81], which is a decomposition of a system architectural description into five views, shown in Figure 3.1. The *customer objectives* view (what does the customer want to achieve) and the *application* view (how does the customer realize his goals) capture the needs of the customer. These needs provide the justification ('why') for the specification and the design of the product. The *functional* view describes the 'what' of the product, which includes (despite its name) functional and non-functional requirements. The 'how' of the product (the technical solution) is described in the *conceptual* and *realization* views. In this way CAFCR is focused on the relation between the customer world and the product. The functional view can be seen as the interface between the problem and solution world. Another dimension in specification and design is the *manufactural* view. This view describes operational aspects of the product

manufacturer (not the operational aspects of the customer that belong to the A view) like preferred way of designing and producing, culture, type and number of employees, et cetera, The job of the system architect is to integrate all views in a consistent and balanced way to get a valuable, usable and feasible product.

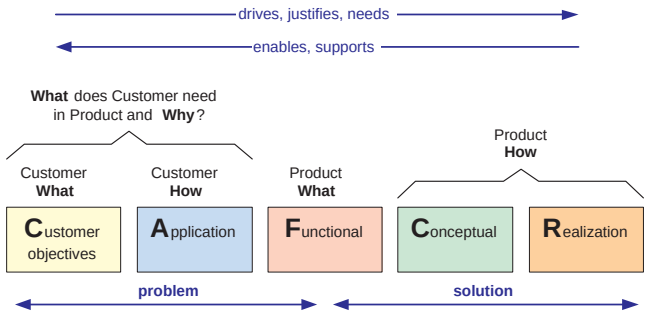


Figure 3.1: CAFCR views

The key driver method can be considered a 'CAF submethod' of CAFCR. The *key drivers* represent the main customer objectives (C view). The key driver method helps to derive the more detailed and quantified *system requirements* (F view). It translates a few (three to six) customer key drivers into maybe hundreds of system requirements. For instance, in the copier case study, 3 key drivers will be expanded to some 40 system requirements. A bridge between key drivers and requirements is a layer of *application drivers*, typically representing the A view. The key and application drivers in the customer views (CA) will be linked via requirements (F) to design choices in the other views (CR). Structuring this information graphically helps to keep a good overview over the design process. It is useful for the designers to see why a certain requirement is important from a customer perspective: to understand the 'why' behind the requirements they have to realize (traceability). The final system requirements are determined on one hand by the customer side (the objectives coming from CA views), but on the other hand just as much by the technical (im)possibilities and solutions side (CR) and the manufactural view. The key driver method focuses on the CAF views, but, as the final key driver model is a living entity, the influence of the CR views grows during the design process. Closely related to the key driver method is Goal Oriented Design [6, 74, 129], which is also based on analysis of the customer needs and goals in a hierarchical fashion. A goal oriented approach like KAOS [74] or GRL [115] starts with goals and derives requirements from them. URN [5] combines use case maps [23, 33, 114] (manufactural and functional requirements) and GRL (non-functional requirements). Also in Quality Function Deployment [96], where the term 'benefit' is used for key driver, the link between benefits, engineering requirements and design concepts is emphasized. The key driver method gains its value from relating a few sharply articulated key drivers to a much longer list of requirements. By capturing these relations, a much better understanding of customer and product requirements is

achieved. The ‘why’ behind requirements is documented and the focus is maintained on the most important issues and the trivial ones are left out. Other important values are its conceptual simplicity and the fact that all product creation phases and corresponding views are taken into account. This chapter describes the key driver method and provides guidelines how to obtain a key driver model. Its application and effectiveness are shown for a high-volume copier. The goal of the industrial case study is to understand the end user requirements by building a key driver model. Moreover, the key driver method is extended by a matrix that links the functional decomposition of the copier to the requirements obtained via the key driver method. This extension offers the possibility to have an overview of the ‘responsible’ functions for realizing a specific requirement as the matrix provides a first breakdown of a requirement into the functional subsystems. This gives a means to monitor the progress or can even be a starting point for further design support techniques, like budget-based design [96]. The outline of the chapter is as follows. In Section 2 the key driver method is presented. Section 3 explains the case study of the copier and applies the key driver method. In Section 4 the final graphical overview of the key driver model is shown and discussed. In Section 5 we combine the results of the key driver model with a breakdown of all the requirements with respect to the functional decomposition of the copier. Section 6 gives the strengths and limitation of the model and the lessons learned. We end with the conclusions in Section 7.

3.2 Key driver method

The key driver method couples the customer side captured in the key drivers (Customer Objectives view) to the product side (Conceptual and Realization view) via the application driver (Application view) and system requirements (Functional view). The following ingredients play an important role:

- *Key drivers* (C view): the top three to six driving customer objectives: what does the customer want? For instance, a copier in a copy shop must print large volumes in a short time. Thus, productivity is a key driver.
- *Application drivers* (A view): Drivers that describe how the customer is realizing the key drivers: how does the customer achieve his goals? Application drivers are closer to the solution space than the key drivers. For instance, the productivity of a copier can be achieved by a.o. speed of printing and reliability. Speed and reliability are typical derived application drivers for productivity.
- *System requirements* (F view): Detailed (often quantitative) specifications of the product or its subsystems. For instance, the speed of printing can be related to the system requirement of 85 pages per minute (for A4 paper) in full production.

The C(ustomer objectives), A(pplication) and F(unctional) views are explicitly included in the key driver model. However, it does not explicitly include the C(onceptual)

and the R(ealization) views (solution side) although these views have a strong influence on the final system requirements as well. Also the operational constraints and preferences of the manufacturer and other stake holders than the main customers determine the final requirements. This information is implicitly included via the process of setting up the key driver model as described below e.g. via interviews with engineers or system architects and using core domain knowledge from the manufacturer's previous projects.

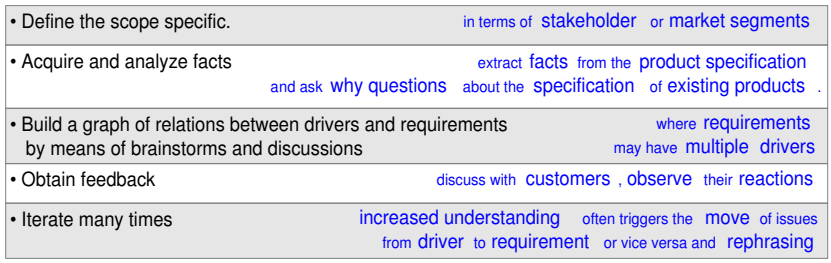


Figure 3.2: Method to link key drivers to requirements by iterating over four steps

Figure 3.2 gives an outline of the key driver technique. The *first step* is to define the scope of the key driver graph. From which customer or other stake holders do we want to understand the objectives or needs? For the choice of the customer it is important to determine the market segment of the product. Also the system boundary plays an important role. For instance, for a copier, do we want to consider a stand-alone copier (with for example an office user as user) or do we consider a complete copy shop (with multiple networked copiers) for which the main operator or the director of the shop might be the main stakeholder? The *second step* is to acquire facts, for example by extracting functionality and performance figures out of the product specification (for the predecessors of the to-be-developed system). Analysis of this information recovers implicit and hidden facts. The requirements of an existing system can be analyzed by asking 'why questions' and mapping this to the new product. For example: 'Why does the copier need additional turning of sheets?' At this point one might have an unstructured collection of various key and application drivers together with requirements. The *third step* is to bring more structure in the facts, by building a graph, which connects requirements to key drivers. A workshop with brainstorms and discussions is an effective way to obtain such a graph. In this case, it is important to get the right people around the table representing the different views: marketing, strategic planning, system architecting and involved engineering disciplines. Also, interviews with people from previous projects or the current development project can be very beneficial in this stage. The *fourth step* is to obtain feedback from customers. The total graph can have a lot of many-to-many relations, i.e. requirements that serve many drivers and drivers that are supported by many requirements. The graph is good if it is as simple as possible and the customers are enthusiastic about the key drivers and the

derived application drivers. If a lot of explanation is required, then the understanding of the customer is far from good. Frequent iterations over these steps improve the quality of the understanding of the customer viewpoint. Each iteration causes some movements of elements in the graph in driver or requirement direction and also causes rephrasing of elements in the graph. The use of the key driver technique benefits from the following guidelines:

- The most important goals of the customer are obtained by limiting the number of key drivers. In this way the participants in the discussion are forced to make choices.
- The focus in product innovation is often on differentiating features, or unique selling points. As a consequence, the core functionality from the customer point of view may get insufficient attention. For instance, consider cell phones that are overloaded with features, but have a poor user interface for making calls. The core functionality must be dominantly present in the graph.
- The naming used in the graph must fit in the customer world and should be as specific as possible. Very generic names tend to be always true, but they do not help to really understand the customer viewpoint.
- The boundary between the customer objectives view and the application view is not very sharp. When creating the graph that relates key drivers to requirements, one frequently experiences that a key driver is stated in terms of a (partial) solution. If this happens, either the key driver has to be split, rephrased, or the solution should be moved to the requirement (or even realization) side of the graph. A repetition of such iterations increases the insight in the needs of the customer in relation to the characteristics of the product. Why, what and how questions can help to rephrase drivers and requirements.

3.3 Case study of a high-volume copier

In Section 1.1 the global functioning of a copier is described. Figure 3.3 presents an overview of a copier together with a decomposition into its major subsystems:

- Scanner module (SCAN): scans hard copy sheets and produces digital images out of it.
- Image processing and job control (CONTROL): generation / adaptation of the digital images coming from the scanner (copy) or network (print) and scheduling of the print jobs (e.g. order of printing).
- Paper input module (PIM): trays from which sheets are separated and sent into the registration module.

- Registration module (REG): paper path that transports and performs accurate positioning of the sheets.
- Print engine (PRINT): transforms the digital information into a toner image which is fused on the sheets.
- Finisher (FIN): collects all finished sheets.
- User interface (UI): the communication means between copier and user.

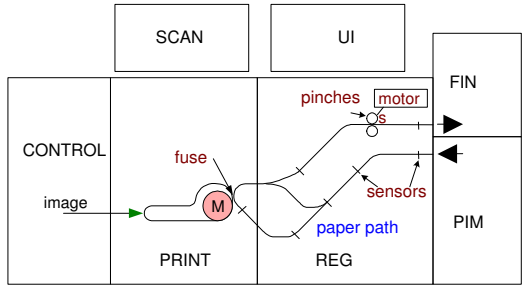


Figure 3.3: Schematic overview of a copier

3.3.1 Step one: Define the scope specific

Stake holders. The high-volume copier under study was aimed at the market segment of the copy shop or the central document production (CDP). The main stake holders are summarized in Figure 3.4. Both the copy shop and the CDP are characterized by a unit (an independent shop or a central place within a company) where individual customers or employees can go to get copies of their originals. Originals vary from simple sheets to entire workbooks. One can distinguish the following customer type stake holders for the CDP (see Figure 3.4):

- Customer: the customer that goes to a CDP to get some copies or prints.
- Operator: this is a professional who uses the system for professional and productive (re)production of end documents. He also plans the work and prepares the jobs.
- Assistant Operator: a professional who uses the system for the professional and productive (re)production of documents. Jobs are processed and planned by the operator.
- System Administrator: an IT-professional responsible to keep the system networked.

- Buyer: responsible for acquiring new equipment.

Next to these stake holders, one also has stake holders for more conceptual, realization and manufactural views:

- Service department: the engineers that service the copier. They come to maintain the copier on a regular basis or in case of malfunctioning.
- Government: this stakeholder states restrictions concerning e.g. the environment (pollution, noise, energy usage, et cetera) and safety.
- Development: the people that create the copier.
- Company board: board of the copier manufacturer that is interested in the business success of the copier.

Selection of the stake holders. Considering all stake holders, some focus is needed. We concentrate on the stake holders of the CDP depicted in the cloud in the top of Figure 3.4, as they represent the customer side.

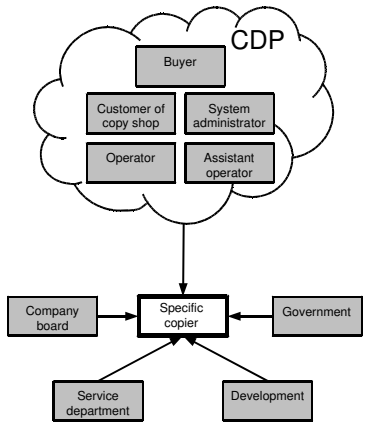


Figure 3.4: Overview of stake holders

3.3.2 Step two: Acquire and analyze facts

For each of the stake holders, scenarios have been generated. Scenarios extend use cases (see e.g. [6, 23, 33, 114]) that are well known in software engineering. In the context of real-time software systems, scenario-based requirements analysis is also used in [99]. Scenarios for complete systems including hardware and mechatronics are very useful to find the essential customer objectives and needs (both qualitatively and quantitatively). The copier manufacturer has a set of such scenarios specified for each

user, which turned out to be very helpful.

Scenarios for specific stake holders. Due to space limitations we cannot present the scenarios for each stakeholder. Instead we concentrate the operator of the CDP. From his perspective, document appearance and finishing quality (reproduction quality, stapling, et cetera) are equally important.

- He is the 'white collar' CDP worker (as opposed to the 'blue collar' assistant operator).
- He is an experienced and eager user of up-to-date repro IT-systems and work flow management tools.
- He prepares, plans, and produces high quality documents, with much variety in the type of documents.
- His main goal is to fulfill the needs of the CDP customer as good as possible. In this respect he is a service provider.
- He uses the system both at the control panel and through specific server workstations.
- His main job is working with the copier as productive and efficient as possible.
- He knows everything about the copier and has been certified by the copier manufacturer. Might be consulted by the copier manufacturer to define new products.

In the first exploration to get to the key driver model, we used scenario-based reasoning using the information above for the operator and we studied the available (public) commercial info and technical information. We combined this with a brainstorm session with people from the copier manufacturer, which gave rise to the first guesses on the key drivers. The initial guess of the key drivers for the CDP, based on expertise and previous projects, was:

- Productivity.
- Print quality.
- Integral cost per copy.

3.3.3 Step three: Build a graph of relations

During the brainstorm session we tried to map the identified key drivers onto the derived application drivers, which in turn can be translated into customer requirements of the copier. This is going from left to right in the CAFCR model in Figure 3.1. Because we experienced that determining the derived application drivers can be hard, we also tried to get them by first making an inventory of the technical system requirements and

then translating them back to one or more derived application drivers. This is going from right to left in the CAFCR model. To refine the initial key driver model, we interviewed the project leader of the development project for this specific copier. The project leader pointed also towards 'ease-of-use' as one of the key drivers. This did not seem to be one of the key drivers for the CDP as there are professional operators in the CDP knowing the machine in all its aspects. As the project leader insisted on the extreme relevance of ease-of-use, we continued the discussion to which customer this was relevant (as we found it inconsistent with the CDP). The discussion led to office users as important end users of the copier, revealing that the copier was aimed at more than one market segments: next to the professional CDP, the copier is also intended as walk-up office copier as found in the aisles of many offices, where people make their own (limited number and limited complexity) copies or print-outs. Hence, the copier seems to have a 'double personality': one product is aimed at two segments. Although this aspect required further attention, this interview confirmed the initial key driver model for the CDP part. To get more insight in the 'double personality' of the copier and to refine the model, we interviewed a person from the business strategy department. Time to market and cost of development were also very important (opportunity and market share). We decided not take these into the key driver model, as they are not directly related to the customer world. These issues are related more to the operational issues of the manufacturer. The business strategy perspective showed (as to be expected) the major differences in priority of the drivers and requirements as compared to the project leader or the end user. Important for us was that the interview with the business unit confirmed the two different market segments (CDP and walk-up) for one and the same copier. As a consequence, it was necessary to split the customers in two groups and include them both explicitly in the key driver model. This required a reconsideration of step 2.

3.3.4 Step two revisited: Acquire and analyze facts

As mentioned in the previous section, it was concluded that indeed an additional branch of customer representatives (related to walk-up) was needed. To include the corresponding additional stake holders, Figure 3.4 is extended resulting in Figure 3.5 that displays the stake holders for the two market segments: the right and left upper branch are corresponding to the CDP and the walk-up environment, respectively.

The additional stake holders for the walk-up copier can be characterized as follows:

- (Generic) Office users: the people that send documents to the walk-up copier electronically to get them printed or go to the machine themselves to get copies of their documents.
- Super Users (e.g secretaries): experienced and heavy users of the copier. Therefore, often asked by other employees for advice.
- Key Operator: a person responsible to keep the system up-and-running. He knows the copier sufficiently to perform day-to-day maintenance.

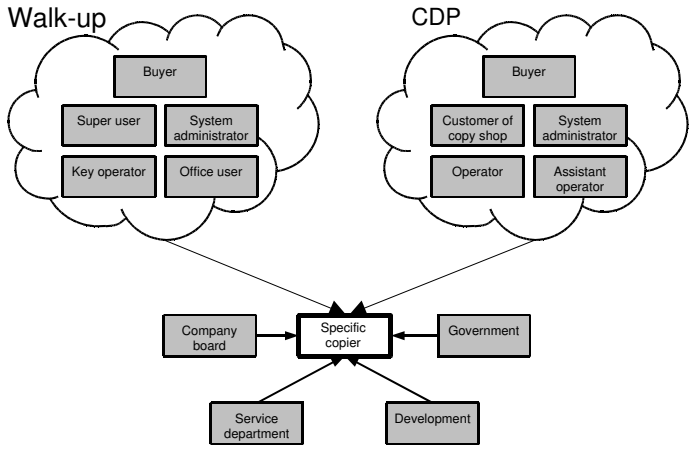


Figure 3.5: Extended overview of stake holders

- The buyer and system administrator are similar as for the CDP.

To give an idea of how the walk-up environment is used, we describe the **generic office worker** in more detail. This stakeholder represents an office worker who uses the system for (re)production of work documents (typically, presence of information is more important than appearance).

- He can have any level of education and task in the office.
- He produces mainly standard documents (A4, straightforward finishing) with almost always the same settings. Typically single or a limited number of copies or print-outs.
- He uses the system both at the control panel and through software on his desktop.
- He works with standard office tools (like windows, office, et cetera)
- He knows how to perform his own tasks on the copier, but not more. He asks the super user in case of problems.

Due to the additional market segment for the copier, the process of reasoning, interviewing and discussing was repeated and finally we came up with the following key drivers (note that we kept their numbers restricted, i.e. three for both market segments). For the CDP we selected:

- Productivity (for large volumes): the CDP needs to print large volumes.
- Versatility: suitable for different kind of jobs.
- (Re)production quality: this is what CDP sells.

For the walk-up copier the resulting key drivers are:

- Minimal waiting time: people do not want to wait (long) for their jobs to be complete.
- Ease-of-use: the (inexperienced) user is not interested in the working of the machine at all and prefers a one (green) button approach.
- Reproduction quality.

Note that the integral cost per copy or the related issues of total cost of ownership / running cost have been abandoned as key driver for the CDP to keep the number of key drivers limited. We opted to focus on the actual users of the copier (with the customer of the CDP and the (assistant) operator as the main stake holders) and less on the buyer who is responsible for the integral cost aspects of the copier. However, integral cost per copy would be added to the key drivers if we would put more emphasis on this type of stakeholder (going to four key drivers for each segment). Typically, the integral cost per copy could be subdivided into application drivers like cost of ownership, costs of consumables, total number of prints (life span), personnel effort (to operate the copier), et cetera, System requirements related to the application driver consumables are for instance, toner usage, service frequency (e.g. related to number of paper jams) and price, et cetera.

3.3.5 Step four: Obtain feedback

At several steps during the development of the key driver model, the results were discussed with people from the organization of the copier manufacturer. The people from the marketing and business strategy were viewed as the internal representatives of the customer side as they are closest to them. Their response on the resulting key driver model was enthusiastic, thereby confirming the correctness and value of the model.

3.3.6 Iterate

In the previous sections only the main part of the whole process of obtaining the key driver model is presented. This already reveals its iterative nature, although many more iterations were made. By iterating over these steps and going from the C via the A to the F view and vice versa many times, a good overview of drivers and requirements was produced.

3.4 Overview of the key driver model

As mentioned, the copier aimed at two rather different market segments. Both sides have different key drivers and application drivers that somehow have to be merged into one set of system requirements. How this is done is reflected nicely in the key driver model in Figure 3.6.

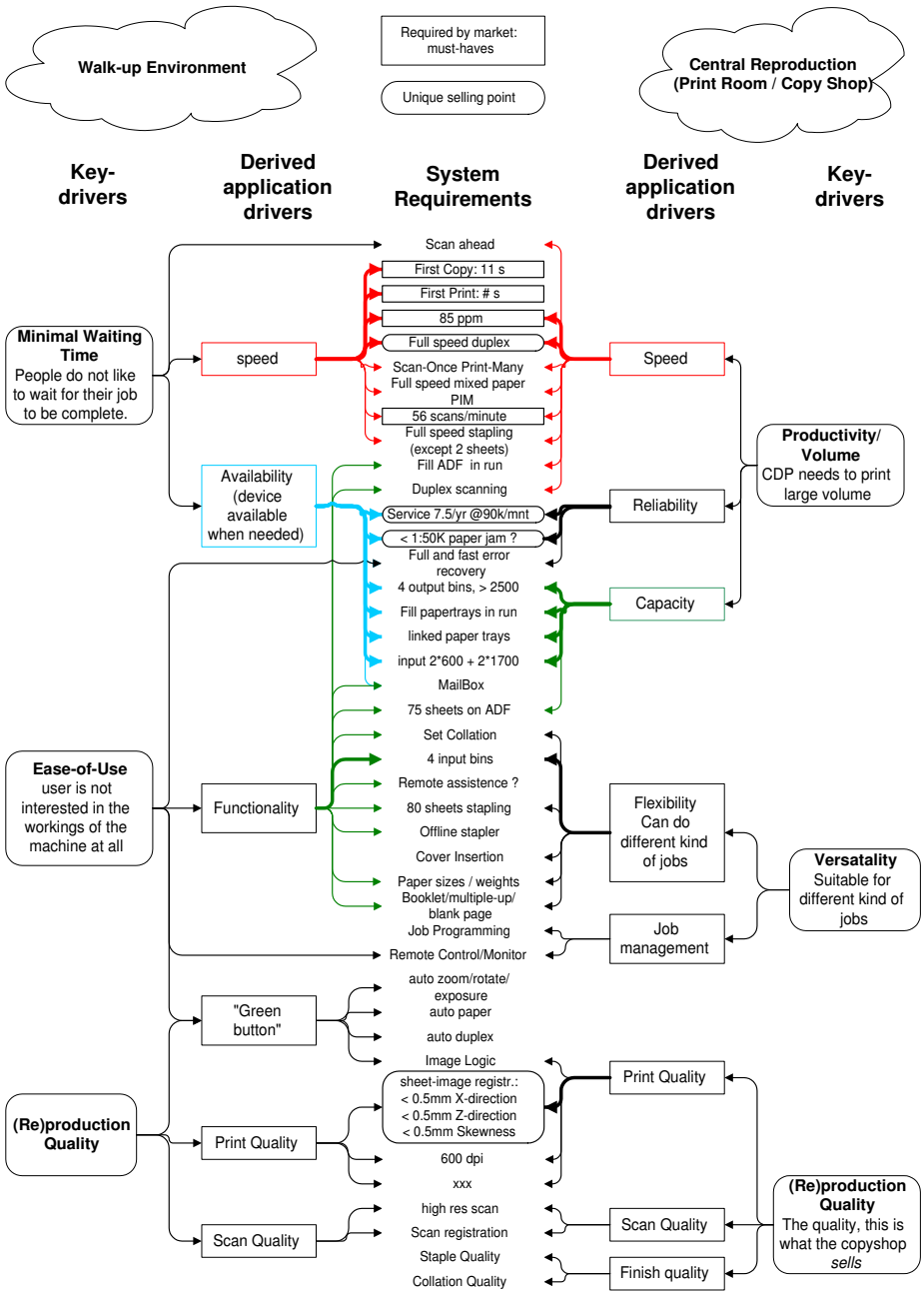


Figure 3.6: Copier key driver model

First, we will give some explanation of the graph:

- Boxes denote the system requirements: Rectangles indicate the must-haves, assets that are indispensable for the market. Rounded rectangles indicate unique selling points if compared to the competitors on the market.
- Thickness of arrows: the thicker the arrow, the larger the influence of a specific requirement on the corresponding application driver. For instance, the CDP-application driver print quality is mostly determined by the sheet-image registration.

To explain the obtained key driver model, we highlight some parts. The *minimal waiting time* for walk-up is coupled to the derived application driver *availability*. In general, a walk-up user gets irritated if the copier is not available irrespective of the reason. Thus paper jams have to be very infrequent. The corresponding system requirement is that paper jams have a frequency of occurrence less than a specific number of printed sheets. Another system requirement related to the application driver availability is a large stock of sheets in the paper input module. In this way the general office user is rarely confronted with empty trays. The very low frequency of paper jams and the huge stock of sheets are considered to have a larger influence on the availability than *full and fast error recovery*. Therefore the former ones have thick arrows coming from the application driver, whereas the latter one has only a thin arrow. There are a number of many-to-many relationships in the graph: the requirement *full and fast error recovery* is related to the application driver *availability* with corresponding key driver *minimal waiting time*, but also directly to the key driver *ease-of-use*. Vice versa, a certain key or application driver is naturally coupled to many requirements. With respect to the double personality of the copier, some observations can be made. Requirements like *time-to-first-copy* and *time-to-first-print* are typically related to walk-up and not to CDP. This phenomenon is directly related to the different key drivers for both environments. Due to the large volumes and the continuous production of the copier in a CDP environment, the initial waiting time does not have a large influence on the overall speed of a job (productivity/volume). However, for walk-up this has a major impact on the speed of the job (minimal waiting time) due to the low volumes. The system requirement *cover insertion* is only relevant from a CDP key driver. A typical walk-up user makes only a few copies and inserting a cover can easily be done manually (otherwise he would probably give such a job to the CDP). For producing high volumes as in a CDP automatic cover insertion maintains a high productivity (for various different jobs). The customer objective here is high productivity of versatile jobs.

3.4.1 Decomposition linked to requirements

Consider again Figure 3.1 with the decomposition of the copier into various modules. This decomposition can be mapped on the key driver model, which results in Table 3.7: the system requirements obtained via the key driver method are mapped to the subsys-

tems of the decomposition - only the first part of the matrix is displayed for shortness. A ‘1’ in the figure means ‘related’ and an empty cell means ‘not related’.

System Requirements	PIM	REG	FIN	PRIN	SCAN	ONTR	UI
Scan ahead					1	1	1
First print out	1	1	1	1		1	1
First copy out	1	1	1	1	1	1	1
85 pages per minue	1	1	1	1		1	1
Full speed duplex		1					
Full speed mixed paper (PIM)	1						
56 scans per minute					1	1	

Figure 3.7: Relating system requirements to system functions

A column gives the system requirements that a certain subsystem has to contribute to and consequently this indicates which requirements the ‘subsystem implementation team’ is responsible for. A row indicates how a requirement is distributed over the subsystems and one can see the responsible and contributing subsystems to each individual requirement. This is very useful for supporting the system design. Next to appointing responsibility, it might also be used as input for budget-based design [96] as one already has a qualitative distribution of a system requirement over subsystems. For instance, time-to-first-print is a shared responsibility of the implementation team for CONTROL, PIM, REG, PRINT, FIN, UI and leads to a rough estimate of the time-to-first-print as the sum of the UI response time, warming-up time (PRINT), and separation / transport / stapling time (PIM, REG, FIN). Note that the duration of the job control and image processing plays a role as well, but good design would aim at performing these tasks (in CONTROL) in parallel with the separation / transport time in PIM and REG. Of interest are columns or rows that do not contain any numbers at all. A row that does not contain any numbers means that a certain system requirement is not related to any module in the system decomposition and hence, the question arises how this requirement will be met. A column that is empty - which is less occurring in practice - means that a certain module is not related to any of the system requirements and thus not related to any of the key drivers. As a consequence, one might severely question its existence.

3.5 Strengths, limitations, and lessons learned

The strengths of the key driver model are the good (high-level) overview it provides of the system. System requirements are linked directly to the key drivers of the customer (view). As such, the key driver model is an excellent means to use in discussion and communication. It is valuable for communicating the goals of a project to the stake holders and moreover, it enhances the understanding between the project leader and his developers. It gives a good means for pointing out why certain requirements are a must and why others are less stringent. A final benefit is its support in making sound tradeoffs between the requirements of a product. Indeed, the model helps tracing the

impact of a change in requirements back to the way in which the customer or other stake holders will perceive the change. After all, satisfying its stake holders is all a product needs to accomplish. One of the limitations of the current model is that not all stake holders have been included. For instance, buyer, service department, and development were not taken into account to keep the focus and the overview. Of course, it is possible to include these in one key driver model (as we briefly discussed for *integral cost per copy* in Section 3.4), but this increases the complexity and possibly more key drivers have to be added. This might cause loss of overview and insight. Other key driver models for different stake holders might be a solution in this case. Building a tool to support the modeling process and visualization (facilitating hopping from one stakeholder to another) would be beneficial. However, the skill is to find a balanced mix of the customers to set up one key driver model that leads to a balanced set of system requirements as in the end only one copier is produced. From this broader perspective, the key driver model of Figure 3.6 would require the extension with key driver *integral cost per click*. To keep Figure 3.6 compact for presentation purposes, we left it out and focused on the actual end users of the machine as discussed before. This is also one of the lessons learned: in order to keep the overview in a key driver model, one has to restrict the number of key drivers, which forces the modelers to make clear choices. Another lesson learned is that scenario-based reasoning from a customer perspective is very helpful in setting up a key driver model. The relevance of scenarios in this context was also pointed out by [99]. The extension of the key driver method to include a matrix that maps the system decomposition to the system requirements is an effective means to support the design. Turning requirements from implicit to explicit has the advantage of clear communication and negotiation. However, the key driver model should be used with the right attitude: in an early design phase requirements might still be up to change and flexibility and openness should be kept. Constant checking a product against requirements is necessary throughout product development. This might require adaptation of the key driver model. It is a *living model* and that is the way it should be used. However, in general one must aim for stability of the (qualitative) model and aim at changes in only the quantitative part during the design process. One has to be careful by just copying requirements from previous projects (*re-use of requirements*). However, an advantage of the key driver method is that, if the requirements for a new product need modification, the method assist in detecting this. Identifying potential new key drivers and connecting them to the application drivers and requirements should reveal if (re-used) requirements need to be altered or removed.

3.6 Conclusions

In this chapter we described the key driver method and presented various guidelines for its use. Its effectiveness was demonstrated by applying it to an existing high-volume copier. The key driver method provides guidance to capture the system requirements and to focus the development. The main advantages are the overview it provides, its conceptual simplicity and the clear visibility of the tradeoffs. In this way it forms a

good means for communication and discussion, especially since it aims at including the requirements that play an important role and leave out the trivial ones. Interesting for the specific key driver model presented here is that the copier aimed to fulfill the needs of two different market segments: walk-up and CDP. This was directly reflected in the key drivers that differ for both: minimal waiting time, ease-of-use and reproduction quality for the walk-up and productivity/volume, versatility and reproduction quality for the CDP. The product itself had to unite these needs via one set of system requirements. Several lessons have been learned and we believe that the key driver method is a useful means to support the design of a copier. Several engineers and architects of the copier manufacturer valued the model. Of course, there is always room for improvement, for instance in requirements elicitation. However, in general we conclude that focusing on the most important issues and providing an overview via a key driver model is a valuable tool in the design of high-tech products like copiers.

Chapter 4

Threads of reasoning

Authors: J.H. Sandee, W.P.M.H. Heemels, G.J. Muller, P.F.A. van den Bosch and M.H.G. Verhoef

This chapter is a re-worked version of the work published in the proceedings of the 16th annual international symposium 2006 of the International Council on Systems Engineering (INCOSE) [101].

4.1 Introduction

The complexity of products being designed by industry today is increasing at an astonishing rate. The search is for a product that will satisfy the design drivers within certain margins. Design drivers are the important system aspects on which design decisions are based. Examples are: development costs, production costs, time-to-market, throughput, response time, productivity, physical dimensions, power consumption, noise production, and so on. Often, design drivers are conflicting, so that trade-offs must be made.

The main need in the design process of a product is to bring structure in the typical chaos of uncertainty and the huge amount of realization options present. This is most profound in the early design phase. Even typical product requirements might be uncertain in the sense that they are only known up to a certain degree or are still open for discussion. Potential solutions or applied technologies all have advantages as well as disadvantages, which causes *conflicts* in the design. A *conflict* is the situation where a specific design choice influences one or more design drivers positively, while influencing others in a negative way. For instance, in the design of a printer one might consider using stepper motors, DC servo-motors or a combination of both for driving the sheets of paper through the paper path. While stepper motors have the advantage of being cheaper (particularly as they do not require expensive encoders and because of their

long lifetime), they are in general less accurate in positioning the sheets of paper. This causes a conflict between the design drivers *printing accuracy* on the one hand and *cost price* on the other. Of course, more design drivers might play a role in such a decision (e.g. size, power consumption, et cetera).

This chapter applies the submethod *threads of reasoning* [81] to find such conflicts in the design of the paper flow control in the printer. The submethod aims at composing a clear overview of how the conflicts relate to the design drivers. As these relations typically involve multiple design drivers, design choices and their consequences, we refer to these relations as *threads*. The submethod is called threads of *reasoning* as the threads typically reveal the *reasoning* applied by the systems engineer. The details of the submethod are presented together with a 5-step iterative scheme on how to create the threads. Once the main conflicts are identified *qualitatively*, a further *quantitative* investigation by modeling and measurements is necessary. The specific model-based investigations are only indicated briefly.

In several communities there are alternative and / or related techniques available to identify the main relations and conflicts in the design of a product. For instance, in requirement engineering and more particular in [123] one uses the term 'problem bundle' that has similar properties as a thread of reasoning. In [123] these bundles are adopted for structuring a design problem at hand and relating this to the solution space. In product line engineering one has methods like Pulse (see e.g. [10]) and in the system engineering community one uses risk management approaches (see [89, Ch. 6]). These techniques create similar overviews, but more retrospective. Threads of reasoning, on the other hand, is applied throughout the complete system design process and in the various design phases. Therefore, the threads are not static, but continuously changing as the design evolves.

Also in VAP (visual architecting process) (see [78, Ch. 2]) and in ARES (Architectural Reasoning for Embedded Software) [65] related techniques can be found which are especially focussed on software design problems. In TRIZ [3] two important concepts are introduced that are also crucial in our reasoning method: formulating the 'ideal' solution to a problem and identifying the conflicts in realizing the ideal product. Quality function deployment (QFD) [98] relates product requirements of the customer to design choices, which, from an abstract point of view, resembles the reasoning used in this chapter. However, a distinguishing feature of threads of reasoning is that it is graph- instead of matrix-oriented. Matrix-oriented techniques have the tendency that the number of relationships easily explodes and one easily looses overview of the essential threads. Threads of reasoning is particularly focused on keeping only the essential conflicts, which we consider an advantage. As a consequence, it is possible to graphically represent the overview of the most important design issues. Moreover, most of the mentioned methods have a tendency to move more towards the customer context and less to the realization aspects. The case study here shows how threads of reasoning can also be used to support conceptual and realization choices of the technical design.

The disadvantage of the explosion of the number of relationships is also encountered in a complementary approach in which one archives the design process including

the conceptual and realization choices [2]. Often the argumentation why a certain choice has been made is included as well. The documentation typically consists of a chronologically ordered sequence of choices with the aim of traceability: how was a certain choice made at some point in time? If some design changes are made in a later stage, one can still apply the reasoning as kept in the archive. In practice creating and maintaining such an archive is often not feasible due to the enormous complexity. This results in a 'tracing' that is not kept up-to-date, with the consequence that its value diminishes. The threads of reasoning technique aims at keeping the essence of the design choices and helps to keep overview.

The outline of this chapter is as follows. In the next section we present the problem statement and put it in the perspective of the multi-disciplinary design of the printer. In Section 4.3, the 'threads of reasoning' submethod is described. In Section 4.4, threads of reasoning is applied to identify the most important conflicts in the case study. This leads to conflicts that require a further study via modeling, measurements or other techniques to obtain a well-founded trade-off. In the same section, we indicate briefly which models have been applied to do the in-depth analysis. In Section 4.5, the conclusions are stated.

4.2 Problem scope

The problem scope of this chapter is the embedded control design of the paper flow through the printer. The *main* (most important) *design drivers* for this part of the design are:

- throughput (pages per minute),
- printing accuracy (positioning of the image on the sheet),
- time-to-first-print (the time it takes before the first sheet comes out of the printer, after pressing 'start'),
- power usage,
- cost price and
- time-to-market.

The first three items of this list are typical performance requirements of the printer. Items four and five are constraints on important resources. The last one, time-to-market, is a constraint that is imposed by the organization. The design should be such that all design drivers are satisfied within certain predefined margins.

For the case study used in this chapter, we assume that the mechanical layout is already given, meaning that positions of (paper transport) rollers, the length and shape of the paper path, et cetera, are known. The design process is in the phase of selecting the control architecture, including:

- Selection of actuators (type and number of motors),

- Selection of sensors,
- Selection of the processing architecture (e.g. centralized versus distributed control),
- Selection of operating system (interrupt driven or time sliced architectures?),
- Scheduling of sheets for print jobs.

To support the design process at this stage, the submethod of threads of reasoning is applied.

4.3 The technique of threads of reasoning

Threads of reasoning is a graph-based, iterative technique to identify the most important conflicts in the design problem and potential solutions. The system architect uses threads of reasoning implicitly to integrate various views in a consistent and balanced way, in order to design a valuable, usable and feasible product. Architects perform this job by continuously iterating over many different points of view and sampling the problem and solution space to build up an understanding of the case. These threads are *made explicit* by the technique of threads of reasoning.

This submethod, as presented in the next section, is based on the work [81, Ch. 12]. A difference between the technique used here and the one by Muller lies in the used *categories*. The categories are the components of the threads, which are coupled through their relations. In particular, threads of reasoning in [81] uses the CAFCR framework that adopts the *Customer objectives* (addressing the *what* question from the customer perspective), *Application* (addressing the *how* question of the customer), *Functional* (addressing the *what* question of the product), *Conceptual* and *Realization* views (addressing the *how* of the product). Instead, it was more suitable in our case to use the following four categories:

- *main design drivers*: limited set of the most important design drivers (typically applying to system level), see Section 4.2,
- *sub drivers*: drivers, derived from the main design drivers (typically applying to subsystem level),
- *design choices*: possible solutions or realizations,
- *consequences*: indicating consequences of a design choice.

The threads themselves are formed by multiple connections between the categories above.

4.3.1 Overview of threads of reasoning

Figure 4.1 gives an overview of the iterative process of the threads of reasoning submethod. Step 1 is to *select a starting point* for the process. After step 1 the iteration

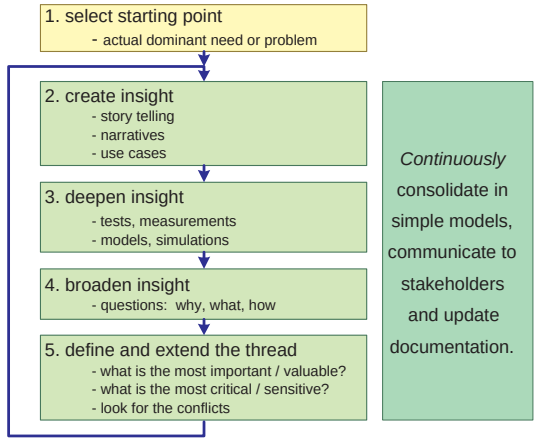


Figure 4.1: Overview of the threads of reasoning approach.

starts with step 2 *create insight*. Step 3 is *deepening the insight* and step 4 is *broadening the insight* via suitable questions. Step 5 *defines and extends the thread*. Moreover, the next iteration is prepared by step 5. In step 5, first the most important and critical threads are selected and one aims at finding conflicts. This insight and refinement might lead to selecting the next need or problem for the new iteration. During this iteration continuous effort is required to *communicate with the stakeholders* (the ones involved in the specific design decisions) to keep them up-to-date, and to *consolidate in simple models* the essence of the problem, and to *update the documentation* to capture the insights obtained.

As mentioned before, the focus of threads of reasoning is to select the critical design issues (step 5) that require in-depth studies to make a sound design trade-off. The in-depth studies are essentially step 3 in Figure 4.1. The limited models for *consolidation, communication and reasoning* are derived from these possibly more complex and detailed models for analysis. Especially, since these in-depth studies require a major part of the design time, one has to be selective in the ones that are actually carried out. Of course, this does not mean that once the answers of these analyzes have been obtained, the thread of reasoning is finished. On the contrary, it might actually be altered based on the findings or continued given these new pieces of information.

Below we will describe each of the individual steps in more detail. Moreover, we will present one thread of reasoning as an example from the case study to illustrate the steps.

Step 1: Select a starting point. A good starting point is to take a need or problem that is hot at the moment, within the problem scope. If this issue turns out to be important and critical then it needs to be addressed anyway. If it turns out to be not that important, then the outcome of the first iteration serves to diminish the

worries in the organization, enabling it to focus on the really important issues. In practice there are many hot issues that after some iterations turn out to be non-issues. This is often caused by non-rational fears, uncertainty, doubt, rumors, lack of facts, et cetera. Going through the iteration, which includes fact finding, quickly clarifies the relevance of the issues.

Example. An important issue in the paper flow control is the question how many processing nodes should be used. Because of the size and the complexity of the software, which is both soft real-time and hard real-time for the various implemented functions, it is almost impossible to process all the code on one node, i.e. one processor. Nevertheless, there are various ways to distribute the software functionality over different (numbers of) nodes. There can be several 'local nodes' that handle separately the control of single motors. Another option is to have only two big processing nodes that handle the entire paper flow control. This design issue is selected as the starting point of the thread.

Step 2: Create insight. In this phase one wants to obtain a rough overview of and insight in the chosen issue. The selected issue can be considered by means of one of the many (sub)methods to create more understanding. Typically, this can be done by the submethods story telling [81, Ch. 11], narratives [33] or scenario-based reasoning using e.g. use-cases [33]. Using these submethods, it will quickly become clear what is known (and can be consolidated and communicated) and what is unknown, and what needs more study and hence forms input for the next step.

Example. To create some first insight into the problem of selecting the number and sizes of the processors in the control architecture, we linked this issue to the main design drivers of Section 4.2. For the time-to-market to be short, it is important to have a predictable development process. Therefore, a concurrent design process is preferred, which is in favor of having multiple processing nodes. On the other hand, we also want the cost price to be low. Here, the question pops up how the cost price relates to the number of nodes. Looking at the design driver power consumption, there is an obvious relation that more nodes require more power, but more specific information is needed to reveal the exact relation and its importance.

Step 3: Deepening the insight. The insight is deepened by gathering specific facts. This can be done by modeling (and model-based analysis), or by tests and measurements on existing systems. Since the presented technique is iterative, in a first iteration one aims at using simple models, measurements or facts that are obtained in a reasonably short time. Typically, back-of-the-envelope calculations or rules of thumb that are known from previous projects are useful. In a second or subsequent iteration one selects the essential issues (most uncertain, most important) that require more modeling and analysis effort. This aspect is coupled directly to the Boderc design methodology [56] based on multi-disciplinary modeling: to discover and select the in-depth modeling activities that have to

be performed to support the system architect in taking (well-founded) design choices. In the design it is important to only spend time on the crucial issues and not on trivial ones to keep both the design effort and the time-to-market limited. Typically, the models are aimed at shedding light on the conflicts, which were identified earlier (step 5, first iteration).

Example. To get deeper insight in the issues of cost price and power usage of processors, more specific information is needed. A rough quantitative estimate for the cost price showed that a node costs typically about 40 euros, of which 10 euros is calculated for the controller and 30 euros for the printed circuit board (PCB). Because for every node a separate PCB is used, doubling the number of processors roughly means doubling the cost price, although the cost price of the processor can be somewhat less for simple variants. Looking at power demands, it turned out that both the smaller and the bigger processors use about 3 Watt. It would therefore be beneficial to have as few processors as possible. On the other hand, if we look at the power demands from other modules in the printer, that use up to 2 kW, we can assume that the power demand from the processors is of minor importance [48]. Therefore, the power issue will not be included in this thread of reasoning as we aim at describing only the most important aspects.

Step 4: Broadening the insight. Needs and problems are never nicely isolated from the context. Therefore, the insight is broadened by relating the need or problem to the other categories. This can be achieved by answering *why*, *what* and *how* questions. Examples: How can a main design driver be realized by sub drivers? How is a certain issue tackled? Why is a certain design choice good for a specific design driver? What are the consequences of a design choice? How is the consequence related to a specific driver? The insight in the main design driver dimension can also be broadened by looking at the interaction with related system qualities: what happens with safety or reliability when we increase the performance?

Example. What happens if all software would run on two processors? An issue that arises almost immediately from this question are possible synchronization difficulties. This is a typical aspect that needs to be considered in further iterations. Other example questions for the case-study are: If we separate the software over multiple nodes, how efficiently can the software still be implemented? How would multiple processors be connected?

Step 5: Define and extend the thread. In the previous steps and corresponding discussion of the needs, design choices and problems, many new issues pop up. A single problem can trigger an avalanche of new problems. Key in the approach is not to drown in this infinite ocean full of issues, by addressing the relevant aspects of the problem. This is done by evaluating the following aspects:

1. Which specification and design decisions seem to be the most conflicting?
2. What is the value or the importance of the problem for the customer?

3. How difficult it is to solve the problem? It is important to realize that problems that can be solved in a trivial way should immediately be solved.
4. How critical is the implementation? The implementation can be critical because it is difficult to realize, or because the design is rather sensitive or rather vulnerable (for example, hard real-time systems with processor loads close to 70% or higher, due to which low priority tasks could be blocked for too long).

To evaluate the above aspects, the system architect often uses 'gut-feeling' based on many years of experience. Analysis techniques, such as Failure Mode Effects and Criticality Analysis (FMECA) can be used to analyze the impact of potential problems in the system in a more structured way. Typically, these techniques are used when the design is finished but they can be equally productive during other life-cycle phases of the design process. To compare various solutions, trade studies [89, Section 11.16] can effectively be applied as well.

The next crucial step is to define the thread. In this step the *important* relations between the design drivers, design choices and consequences are represented in a concise diagram. Furthermore, the important conflicts should be clear from the diagram. The problem, that serves as the starting point for the next iteration, can be formulated in terms of this conflict. We believe that a clearly articulated problem is half of the solution.

The insights obtained so far, in terms of the most crucial and critical conflicts, should help to select the new need or problem to go into the next iteration (back to step 2).

Example. At this moment in our reasoning on the number and size of processing nodes, the first thread becomes visible, as visualized in Figure 4.2. The thread is structured by means of the framework of the categories as introduced before. The interpretation of this visualization is as follows:

- On the top of the picture, the relevant *main design drivers* are given in capitals,
- From the main design drivers, *sub drivers* are derived, indicated in bold face,
- Specific *design choices* that satisfy the sub drivers are indicated in italic,
- The *consequences* that come with specific choices, are depicted with small dashed arrows,
- The main *conflicts*, that are identified between any of the above mentioned aspects of the system, are depicted with thick double arrows.

Note that in step 3 we already concluded that the main design driver power should not be included in this thread. Hence, a step 5 action of discarding less relevant aspects of a thread was already applied. We see that from the question of how many processing nodes to use, a conflict arises between the drivers 'time-to-market' and 'cost price'. As the most profound conflict is identified now, this

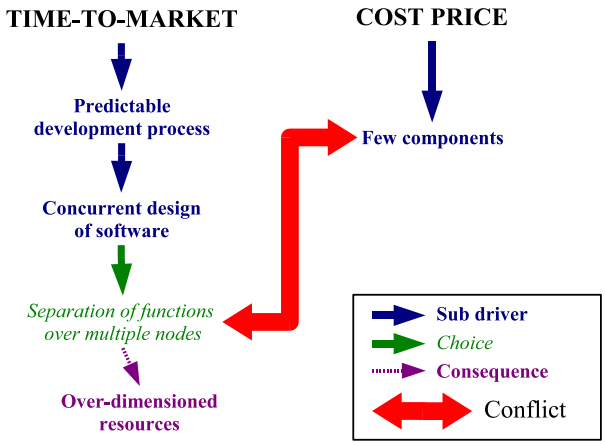


Figure 4.2: Example thread in the design of the paper flow control.

can be input for step 2 and subsequently step 3. More detailed models (in comparison with the simple estimates of cost price done earlier) would be useful to deepen the insight, which would support in making the trade-offs in the early design phase. From our first simple models we concluded that for reasons of cost price we want as few processing nodes as possible. However, a proper software design should still be feasible within a limited time span (influencing time-to-market). Therefore, we used a Parallel Object Oriented Specification Language (POOSL) model [95] (see also Chapter 13). With this modeling language and analysis techniques, several possible architectures are evaluated and compared on their feasibility with respect to software timing requirements. Note that a part of the argumentation of a particular choice is captured now in the specific models made. In another setting (or a different architecture) this can be used to reevaluate the design choice. So some kind of 'tracing' - as discussed in the introduction of this chapter - is kept.

The thread of reasoning of Figure 4.2 was obtained by iterating one-and-a-half times through the 5-step scheme of Figure 4.1. As we will see, this is typical for the case at hand as the aim of threads of reasoning in this setting is to select the in-depth models to be made. Normally more iterations are used to find the essential conflicts for instance, continuing after the modeling step.

4.4 Threads of reasoning for the case study

The structure that covers the most important threads and their relationships can be complicated for the design of complex systems, like a high-volume document printing

system. In addition to the thread presented previously, we will describe two other essential threads in the control of the paper flow. In the figures below we will use the same interpretation of the visualization as in Figure 4.2.

4.4.1 Stepper motors versus DC servo-motors

In this second example thread, the starting point is the use of stepper motors instead of the originally used DC servo-motors for driving the rollers in the printer paper path. The use of DC servo-motors is common for the printer manufacturer and less experience with stepper motors is present.

To create insight (step 2), the use of stepper motors was related to the identified main design drivers. It was easy to see that stepper motors relate to the cost price of the system, as the reason to select them in the first place was the fact that they are cheap. DC servo-motors are more expensive because of their need for (expensive) encoders and shorter lifetime. The use of stepper motors also relates to the printing accuracy. The accuracy of a stepper motor is limited because of various reasons, such as its mechanical construction, cogging and overshoot [47]. Because the stepper motors have to control the movement of the sheet, the sheet can only be controlled with limited accuracy. With a DC servo-motor (in combination with an encoder) the movement of the sheet can be controlled up to much higher accuracy and therefore is no issue.

To see whether the aspects discussed above are really important, we need to deepen our insight (step 3); in this case by quantifying the reasoning. The first aspect was the cost price. The average price of a (low power) stepper motor does not differ that much from the average cost price of a DC-motor. Both can be obtained (for large quantities) for typically less than 10 euros. For both types of motors an electrical driver is required, which also costs about the same for a stepper motor as for a DC-motor, i.e. circa 3 euros for low power applications. An encoder, which is solely needed to control the DC-motor, cannot be obtained below 20 euros for high resolution rotary encoders. This is one of the main reasons why the use of stepper motors is preferred.

Another aspect that needs some quantification is the accuracy of the stepper motor. First measurements reveal that this indeed is an important issue. Figure 4.3 shows a plot of position against time of a stepper motor running at 1 rotation/sec. Four steps are visualized of a 200 steps/revolution motor. The dashed line corresponds to the reference position, the solid line to the actual measured position. The horizontal grid lines indicate the size of the four steps that are visualized. Each step of the motor can be translated to a step-size in the order of 0.2 mm of the paper. From the figure it can be seen that the inaccuracy in the motors position is about 1 step size, i.e. 0.2 mm. As the printing accuracy is defined at 1 mm, the paper needs to be positioned with an accuracy well below 1 mm. The obtained value of 0.2 mm is therefore critical and needs to be evaluated further. It is nevertheless hard to quantify the impact on the real position of the sheet, because of load differences, the occurrence of slip and interactions between two motors that are controlling the same sheet of paper for some period of time. Therefore, more extensive models are needed.

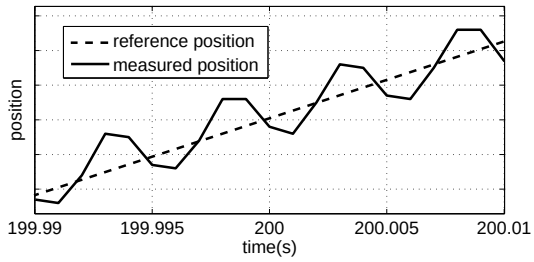


Figure 4.3: Measurement result of stepper motor.

Note that the above reasoning illustrates the typical back-of-the-envelope calculations that quantify the reasoning.

Like in the first example thread, we broaden our insight by means of the *how*, *what* and *why* questions (step 4). The first question could be how the motor should be controlled. The answer to this question is that a frequency generator needs to be implemented as for every step of the rotor, a drive pulse is needed. The follow-up question to this answer is how this frequency generator could be implemented. This pinpoints the question whether to do this with dedicated hardware or in software. Note that this question is a common struggle in industry nowadays. It comes down to the question whether cost price or accuracy and predictability is more important. Normally, hardware implementations are more reliable and faster or more accurate, but increase the cost price of the system.

The last step in this first iteration is the visualization of the thread. This is depicted in Figure 4.4. We see that two important conflicts have been identified that need more attention. The first one is the use of dedicated hardware for the frequency generator in relation to the use of few components to reduce the cost price. The second conflict is identified between the limited accuracy of stepper motors and the requirements on the control accuracy of the sheets.

4.4.2 Time sliced versus event-driven architecture

During the design, a time sliced architecture was proposed for the processing nodes on which, for each node, multiple tasks are scheduled. The idea is that by assigning each task its own time slice, the execution of different functions is temporally separated and task interference is thus avoided. Therefore, software functions can be developed and tested separately while guaranteeing that it will work after combining them on one processor if each task fits in a slice and there are enough slices. The fact that this choice also has some important disadvantages, makes it a good starting point for a new thread (step 1). To create insight (step 2), we again relate the issue to the main design drivers. The main reason for adapting the time sliced architecture is to shorten the time-to-

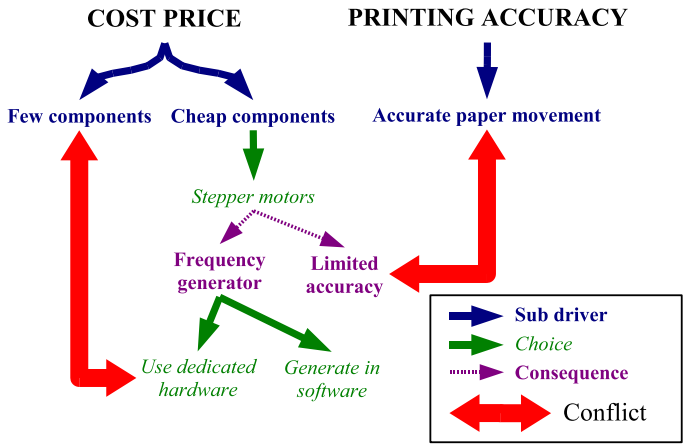


Figure 4.4: Thread of the example of stepper motors.

market, as it enables predictable and composable software design. Furthermore, we can use existing knowledge from past experience of the printer manufacturer (since the time sliced architecture has been applied in the past).

One of the disadvantages of using time slices is the inefficient use of available processing power. Because each task gets a pre-determined part of the available processor time, tasks cannot use the slack time of each other. To quantify the inefficiency of the time sliced scheduling in our case (step 3), we created a simple spread-sheet model which shows the tasks, the expected processor usage and the size of the slices. It also includes an estimation of the interrupts that can occur. Because the interrupts can interrupt any task, a task can effectively take longer to execute than its measured execution time (without interruption). To guarantee the composability of the system, we have to take this interrupt overhead into account for every slice. It turned out that the overhead of the interrupts in a time sliced approach is 20%, while if we replace the time sliced approach by e.g. a rate monotonic scheduler, it becomes much less: 3%.

To broaden our insight (step 4), we could ask ourselves what the influence of the choice of the time sliced architecture would be on the printing accuracy. From past experience, but also from literature it is known that the time sliced architecture introduces a limited action-reaction speed. As we need tight paper-image synchronization for accurate printing, this choice does influence the printing accuracy and therefore needs further in-depth investigation (via modeling).

Figure 4.5 visualizes this thread, together with the first two example threads. From the analysis above, two conflicts are identified between the use of the time sliced architecture (because of the main design drivers: time-to-market, cost price and printing accuracy).

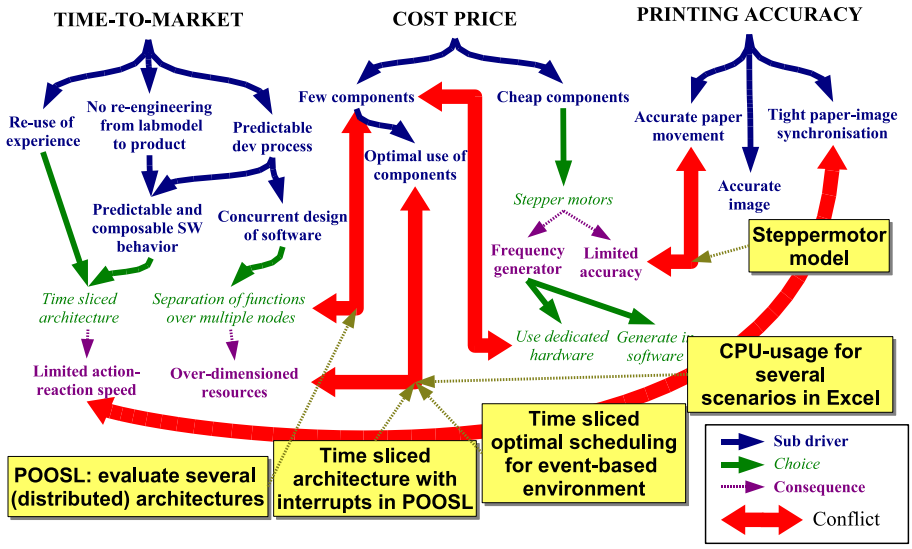


Figure 4.5: overview of several combined threads of reasoning.

4.4.3 Total overview

The three example threads are visualized in Figure 4.5 in one overview graph. It is interesting to see how these conflicts relate to each other. One example is found in the printing accuracy. The requirement of a high printing accuracy not only conflicts with the use of stepper motors, but also with the use of a time sliced architecture.

With the global overview we have obtained a clear list of conflicts where multi-disciplinary models can be made for deepening the insight (step 3). In Figure 4.5, the light grey boxes are added to indicate the models that have been made. These models give more insight into the identified conflicts. As mentioned before, the threads of reasoning obtained here originate from one-and-a-half cycles through the 5-step scheme to end up with the in-depth models to be made. Although Figure 4.5 originates from a limited set of starting issues, related to only a subsystem of the complete printer, and from only one-and-a-half iterations, it already shows a quite complicated structure. Nevertheless, the overview already captures the most important conflicts in the design of the control architecture for the paper path.

4.4.4 Detailed models to obtain insight in conflicts

To deepen the insight, specific models have been made, especially at design considerations where conflicts are identified. Figure 4.5 shows the objects of study of the models in the light gray boxes. To obtain more insight in the conflict explained in Sec-

tion 4.3 (the size and number of processing nodes), a POOSL model is created [95]. With this modeling language and the analysis techniques, several possible architectures are evaluated and compared.

A second model was made in the language POOSL to analyze the processor load for the scenario in which the time sliced architecture is 'polluted' with interrupts, which are necessary to make optimal use of components. This is a more detailed model than the spread-sheet model described in Section 4.4.2. Both models can also be used to see what the consequences are when the frequency generators for the stepper motors are implemented in software.

To optimally use the processors (and minimize the number of processors), a model was made to calculate optimal schedules for tasks in a time sliced architecture [8]. A stepper motor model, created in Matlab/Simulink, was used to analyze the positioning accuracy of stepper motors [47].

4.5 Conclusions

In this chapter, the submethod of threads of reasoning was applied to identify the most important conflicts in the multi-disciplinary design of the paper flow control of the printer. This submethod aids to structure in the typical chaos of uncertainty and the huge amount of realization options present in early design phases.

Threads of reasoning is one of the submethods used in the (Boderc) design methodology that aims at using multi-disciplinary models to predict system performance in an early design phase, while respecting the business constraints of available man power and time-to-market. The restriction in available design time (related to time-to-market and available man power) implies that in-depth and often time-consuming modeling and analysis should be performed only for the essential and critical issues. Threads of reasoning has turned out to be - at least in the case of designing the control architecture for a printer - an effective means to find these issues and to create overview.

Combined with the in-depth models, threads of reasoning provides the system architect with valuable insight that supports him in making the important design trade-offs and to reduce some of the uncertainty in the early design phase. It results in a concise picture with the important conflicts depicted *explicitly*. It forces the designer to quantify choices by replacing hand-waving with facts. This stimulates and focuses the discussion with the consequence of a shorter time-to-market and a more predictable design process. Moreover, a part of the argumentation of a particular design choice is captured now in the specific models made and techniques used.

It is a true observation that threads of reasoning itself does not *create* knowledge. It stimulates to make existing implicit knowledge explicit and aids in discovering which knowledge is lacking and where development time should be invested. In this line of reasoning, one could argue whether a submethod like this is part of an engineering discipline. This does, nevertheless, not diminish the value of the submethod.

Based on the case study, the following suggestions for the use of threads of reasoning can be given:

- Keep the number and the size of the threads limited by selecting the most important ones to keep overview and not to drown in details. In our case study the entanglement was much larger in a first instance of Figure 4.5. Additional iterations were used to regain focus and gave rise to Figure 4.5 in its present form.

- Whether or not certain conflicts are important, depends, amongst other things, also on the level of the system design. The higher the level, the less detail has to be taken into account. Often though, some iterations will have to go quite deep in a short time to gather some facts that influence design choices at a much higher level. It helps to quantify things (even if the numbers might be uncertain in an early design phase) as it sharpens the discussion and replaces 'gut-feeling' by facts. In particular, back-of-the-envelope calculations, figures-of-merit and rules-of-thumb help to identify the essential conflicts and to discard the unimportant ones.

- In the reasoning process, fast exploration of the problem and solution space improves the quality of the design decisions. It is important to *sample* specific facts and not to try to be complete. The speed of iteration is much more important than the completeness of the facts. Otherwise the risk is to get stuck within one particular aspect. It is often sufficient to know the order of magnitude and the margin of error for the trade-off analysis (especially in early design phases). Be aware that the iteration will quickly zoom in on the core design problems, which will result in sufficient coverage of the issues anyway.

- It is essential to realize that such an exploration is highly concurrent; it is neither top-down, nor bottom-up. It is typically viewpoint hopping and taking different perspectives all the time.

We applied thread of reasoning to a relatively simple case study, compared to for instance the design of a complete aircraft. To abstract up to more complicated systems, one can apply thread of reasoning recursively on various levels of detail, for the system and subsystem design. In the example of an airplane, one could start with applying threads of reasoning to the overall design, restricting oneself in not taking too much detail into account. Separate threads can then be created of the various decomposed parts of the airplane, such as the motors and the navigation instruments. In the example of the printer, we could have created a separate thread of the image processing and corresponding hardware up to a less detailed thread for the complete system.

An open question still is how to learn the 'skill' of threads of reasoning. Being able to iterate fast through the design space and views seems to be hard and tends to be driven by knowledge and experience. Making the trade-offs in little time seems to be a skill that you can only learn by experience. However, the guidelines given in this chapter and the presented examples in the case study provide a first step in recognizing the skills needed and, succeedingly, mastering these skills.

Chapter 5

Budget-based design

Authors: H.J.M. Freriks, W.P.M.H. Heemels, G.J. Muller and J.H. Sandee

This chapter is a reworked version of the article 'On the Systematic Use of Budget-Based Design' presented at the 16th annual international symposium 2006 of the International Council on Systems Engineering (INCOSE), Orlando, U.S.A., July 2006.

5.1 Introduction

According to the dictionary, a budget is defined as an estimate of income and expenditure, for a specified period of time. Although financial budgets and overviews are well-known to most people, the usage of budgeting in technical designs is less common. A technical budget primarily focuses on a resource that is considered important in the design or in normal operation of the final product. Technical budgets typically concentrate on distributed quantities (i.e. quantities that are used / shared by various system components) such as standby power, processor load, product size, memory size, response time or accuracy.

In this chapter we claim that using a more systematic approach towards budgeting is an effective means for supporting a technical design process.

5.1.1 Related research

There are various application domains in which technical budgets are used. One of these areas is optical data communication. In this field, optical link-loss budgets are frequently used to check whether enough optical power is available to communicate between transmitter and receiver. An example for an optical budget can be found in [79].

An application field in which budgeting is also widely accepted is lithography. [125] shows how a company compared its overlay budged to the technology roadmap of their industry. In this way they were able to show the evolution of their own equipment in the context of general market trends. In [107] an overlay budget was used to specify the total alignment accuracy of a new type of lithographic device. It is interesting to see how the authors have formulated a number of summation rules that intend to attribute budget parameter values to the corresponding use-case scenarios of the device in question. In this chapter, we will also present an example from the lithography industry.

Also in the area of hardware-software co-design budgets play an important role. An example of this is, for example, given in [9]. The focus in [9] is on making tradeoffs in the timing aspects of how software can be mapped onto different choices of hardware configurations. Budgets of worst and best case execution times of software tasks on various platforms are indispensable in their approach. Also the work of [122] is of interest in this particular application domain.

Another example of budget-based design is given in [68]. This paper shows how a budget was used for designing a processor in such way that the total electrical noise contribution of all subparts could be kept within the specified limits. According to the authors, this kind of planning in the early stages of the design was essential to meet the requirements in a timely and cost-effective manner.

Space agencies also use budgeting in their design processes. In [45] mass budgets are used to regulate the mass of a spacecraft, during development. The total mass of the spacecraft and its load is important as it influences the amount of propellant and the amount of steering force that are needed for manoeuvering through space. Moreover, in [36] monetary budgets are used for setting up complete aerospace programs (unmanned space programs) to minimize the risks of project failure.

Although several case studies on budget-based design have been described in literature in various application domains as outlined above, an overview of the budgeting technique itself, and of guidelines towards using budgets, are hard to find. This chapter aims at filling this gap by taking a more *generic view* on budget-based design and removing the application domain and *resource specific* aspects. Two examples illustrate the general overview and the guidelines.

5.1.2 Influencing the design process

Budgeting basically is the process of gathering and structuring information about a resource that plays a major role in the design, and distributing this resource in the best possible way over a decomposition of the system at hand. That is, once the retrieved information is combined and represented in a clear and distinct manner, it is an excellent means to support the design making process for the distribution of the resource. A technical budget can be made on many topics like standby power, memory usage, processor load, accuracy, or other product-specific aspects. Figure 5.1 depicts the way in which budgeting is supposed to influence the engineering process.

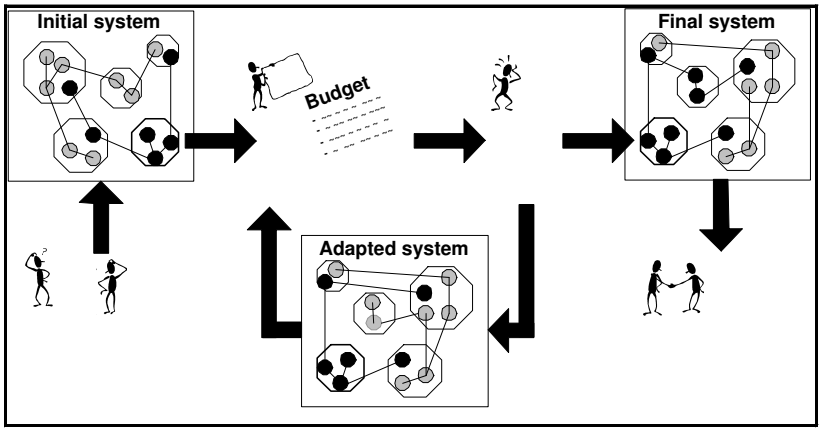


Figure 5.1: The influence of budgeting

In first instance, engineers will start thinking about the design of the system. Using the initial design blueprint, the first budget is created from the information that is already available. During this budgeting process, engineers first select a suitable decomposition of the system and try to quantify the chosen components in terms of resource usage. Experience from previous projects, from measurements, from models or from data sheets is used for this. Eventually this must lead to a budget that expresses a satisfactory distribution of the budgeted resource. In order to adapt the design to meet the specifications, the relationships between design alternatives and resource consumed should be known.

This distribution will certainly raise a lot of discussion between people involved, solve a few misunderstandings or lead to some feasibility studies being started. The relationships mentioned above are used to change the design. Once engineers have agreed upon a new distribution, the initial system is adapted, after which an update of the budget is made. This is an iterative process that continues until a final design is found, which is satisfactory to all parties involved. This means that the design must also fulfill a number of other important system requirements.

Using budgets in a design process serves the following purposes:

- To make the design more explicit.
- To provide a baseline for taking design decisions and verification of the implementation.
- To specify the requirements for the detailed designs of the components.
- To have guidance during integration.
- To manage the design margins explicitly.

Although often assumed otherwise, making a budget *does not* consume a lot of time. Practical cases have proven that, when relevant data is readily available, drawing-up a budget can sometimes be done within a day. This is not a large expense considering the fact that budgeting helps identifying design risks, stimulates faster design and creates commitment between the engineers involved.

5.2 Budgeting

A budget contains the following elements:

- the budgeted resource,
- a decomposition and
- the distribution of the budgeted resource over the decomposition.

Budgeting work starts after one has decided what the subject of the budget will be. In principle, all resources in a design could be covered by separate budgets. However, as the amount of effort to be spent on budgeting is limited, one has to focus on the most important aspects of the design: these aspects can determine success or failure of the whole project.

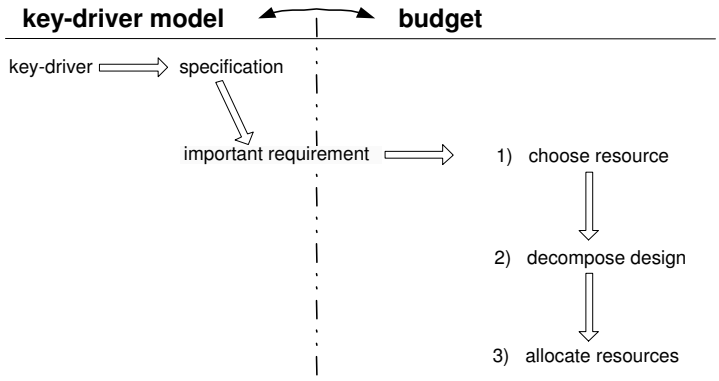


Figure 5.2: (Pre-)budgeting processes

Figure 5.2 shows that the key-driver model can be useful for retrieving those system requirements that are a candidate for budgeting. The key-driver technique as explained in Chapter 3 is a way to determine the main business drivers of a product, from the perspective of its stakeholders, and to relate them to the most important system requirement(s). Of course, one can also use a different technique to identify the most important system requirements. See [74], [99] and the references therein. Ultimately, the resources that are selected to be budgeted are those that are most directly related to

these essential requirements and drivers. The specification / requirement for a resource provides an upper bound for a budget, that should not be exceeded. For instance, if a power consumption budget is made for a device that uses power from the wall socket, the total resource distributed in the budget should not exceed the maximum power available from the socket (which depends on the country of residence). As Figure 5.2 also shows, there is no clear boundary between where the key-driver technique ends and where the budgeting method starts. For instance, while making the budget it sometimes turns out that the budgeted resource has been wrongly chosen or that another requirement plays a much more important role. At these moments, one just has to adapt the existing budget or restart with a new one. The effort that was spent now seems like lost time, but note that one has now eliminated already one of the (alleged) critical system requirements. In the end, this effort will benefit the design.

5.2.1 A systematic approach

The following paragraph shows the basic guidelines for making a technical budget. These budgeting rules can be applied as soon as the topic/resource of the budget has been selected.

Clearly define the scope within which the budget is applied.

Budgets generally have a limited scope within which they are valid. For this reason, one needs to set the system boundaries before starting to budget. First of all, one needs to decide what parts of the design will be included in the budget and what parts will not. For example, some add-on modules may be regarded as a part of the entire product and thus as a part of the budget, whereas others may not. In case of a copier, for instance, the sheet finisher must be connected to a separate wall socket (by design choice). Hence, this part of the product should not be included in the power budget for the copier, either. Furthermore, one has to choose in what operating modes or in what use-case scenarios [33] the budget is valid. A system architect, for example, must decide whether he will be budgeting the typical usage of a product or whether he rather sees the budget based on worst-case conditions, like for instance extremely intensive usage of the device.

Select a decomposition of the system-under-design that suits the budgeted resource.

A *decomposition* is the foundation of a budget. No universal recipe exists for the way of decomposing a design. The *constructional decomposition* and the *functional decomposition* are frequently used for this purpose. From project management point of view, a decomposition that easily maps on the development organization is preferred. This way every component from the decomposition is easily assigned to a responsible person who collects the information and who can be held responsible for meeting the

specifications as agreed. Moreover, it is also preferred to make use of budgets for existing products to obtain an initial decomposition. These budgets have a certain proven authority and therefore provide a first guidance to fill-in the new budget. Furthermore, it is recommended always to stay focused on the budgeted resource. Although people are often inclined to decompose a design into the most obvious functional blocks, the decomposition that most closely corresponds to the budgeted resource is the one that ought to be chosen. Of course, one has to realize that for the project management reasons mentioned before, this might have drawbacks. As such, it is of importance to carefully select the decomposition.

Find the quantitative figures in each of the chosen components from the decomposition.

If available, budgets for existing products provide the first guidelines for filling in the budget for a new design. Although the budget for a new system can be based on an existing one, it must have a number of explicit improvements in order to fulfill the new specifications. The figures that are presented in the budget must be substantiated by means of calculations, estimates, data sheets or simulation of the designed components. Known uncertainties in the used figures and expected margins in the given estimates should always be included. The further the design process evolves, the better the initial figures can be verified with the real design and the better they can be matched to measurements of existing (sub)systems. A budget can thus continuously be improved. Also see Section 5.2.2.

Combine the gathered information into a clear and simple overview and reiterate between the previous steps.

A clear overview must reveal the essence of the budget and the most critical issues to the system architect and his stake holders, in the glimpse of an eye. The architect must ensure the *manageability* of the budgets. A good budget has (at most) tens of quantities described. The danger of having a more detailed budget is loss of overview. An unambiguous representation, like a graphical decomposition, supports the discussion about matching the design to its specification. These debates lead to fruitful negotiations over the design and to the design improvements needed to meet the specifications. This is an iterative process over the previous steps.

5.2.2 Gathering the relevant data

Aside from the basic guidelines as given in the previous paragraph, there is also the art of retrieving relevant data. Although this information can come from various sources, the first source lies within the company itself. As most companies make products with which they are already familiar, experience such as measurement data, design specification documents of previous projects or existing budgets can provide a valuable reference for future designs. This work can be used in order to gain insight in what is

possible in new projects and in order to obtain initial estimates. For instance, data from other devices in the same product family can be interpreted in order to make realistic estimates for the resources required for the new product. Besides that, (measurement) data from reuseable modules can often be extrapolated into figures that are relevant to the current budget. Talking to experts is another valuable source to get estimates. Moreover, predictions and estimates made by experienced colleagues often give a good indication of what can be expected in the new design. Even more so, models, simulations and back-of-the-envelope calculations bring the information one needs.

In addition, also be sure to use the specifications given by third party vendors. If these resources still do not give enough information to complete the budget, one can also turn to articles, browse through academic research reports or even scroll the internet. One could even decide to apply trend analysis (e.g. Moore's Law) or to take notice of the strategy of one's competitors.

5.2.3 Continuous evolution

Projects in industrial environments usually show a fast progress. For this reason, design budgets have a very evolutionary character. A budget is a 'living entity', which has a limited time of validity before new content updates are required. In addition to this, there can be other factors which outdate a budget. The following paragraphs deal with changes that often occur.

Direct Changes. Direct changes are mostly due to changing requirements concerning the budgeted resource itself. An example that played a major role in the development of a digital copier is power consumption. In the initial stage of the design, the requirement on the budgeted resource was set by the power available from the wall socket. After a while, however, people realized themselves that also some work needed to be done in order to make the device compliant to the major energy criteria. Although this change was not initially foreseen, in the end it did have an effect on the power budget as this became tighter (the upper bound on the usage of power was more strict.) In this case the change is *direct* in the sense that it is the power requirement itself that has changed.

Indirect Changes. Indirect changes are design changes that are not directly related to the budgeted resource, but which have an influence on the budgeted resource indirectly. For example, in the development project for a copier system, it was decided in a late stage that the total outer dimensions needed to be some 20% smaller. This does not have a direct influence on the budgeted resource (power consumption) in contrast with a 'direct change' like lowering the maximally allowable power usage. However, the change of the size of the machine required major modifications in the size and shape of the paper path, the number and types of motors used, the schedule of sheets, et cetera. These changes *do* affect the power consumption and thus the effect is more indirect. In the selected case we could fortunately show that the change only had limited consequences on the total power budget. By exactly showing the consequences, we could remove one of the worries for the strategic change in size. From a higher abstraction,

what happened here is that two or more budgets were influencing each other. The size budget and power budget have an effect on each other. Certain design decisions might have a positive effect on one budget, but a negative on the other. The decision making process for the overall design can benefit from using both budgets as this makes the tradeoff *explicit*.

Another example of an indirect change is given by the choice for the type of motors in the same copier project. DC motors were initially assumed to take care of paper transport. Once the first version of the budget was available, some engineers wanted to replace DC motors by stepper motors, for cost price reasons. Due to this design change, the magnitude of the power peaks changed, thus leading to a change in the power budget. In the studied case, the updated budget was used to assess whether this change in motor type could be realized, considering the given resource.

5.2.4 Margins, uncertainties & inaccuracies

One important aspect of filling in a budget is how to deal with measurement inaccuracies and uncertainties. Of course one could stay on the safe side and always assume a worst-case situation. However, this may be much too conservative, causing a feasible design to be judged infeasible instead. As a consequence, one will start improving the design that was wrongly labeled 'infeasible', which can possibly lead to a higher cost price or other negative effects. Dealing with uncertainties in such a way that one obtains a nearly optimal - but still properly functioning - solution, is therefore a true art. Moreover, possible design improvements that require further investigation can be explicitly included in a budget as (obtainable) design margins. These design improvements can be exploited to reduce the total resource usage and thereby matching the specifications better. Of course such improvements should not interfere with other design criteria.

The impact of deviations. One of the factors, which influence how errors should be dealt with, is their impact on the budget, and therefore on the design. Suppose you have conducted a measurement of peak power consumption, which shows that the measured peaks have a mean value of 1.5, a standard deviation of 0.5, but that there is an occasional peak with an absolute maximum of 4.5. Depending on the impact of this peak on the total budget, we have several ways to deal with it. If an occurrence of the worst case peak will cause a severe malfunction of the product, then the value 4.5 needs to be included in the budget. When its impact is less prominent, then one could include the measured value including for instance twice the standard deviation in the budget. If the error is insignificant enough, one might consider leaving it out altogether and use the mean value of 1.5. However, remember that the significance of an error depends on how much margin there is left in the budget, and be aware that many small errors can still make a big one after all. So, explicitly keeping track of errors and margins remains important. Note that this issue is clearly related to step 1 in Section 5.2.1: the interpretation of the figures is related to the scope of the budget (its use-case).

Correlation of deviations. The second factor that needs to be considered is the correlation between deviations in budget figures. In case deviations in the figures are

related, special care should be taken not to handle them as individual and insignificant variations. For instance, imagine that the power peak for a number of components always coincides. The deviation for each component may be insignificant to the total budget, but if they are taken together they might indeed be of influence. In case the relationships between figures are known in more detail, one can also use stochastic techniques to express their influence on the total budget. This actually means that one has a conceptual model of the decomposition and the system-under-design that provides the necessary insight to correctly attribute the overall effect of components to the resource. A good example for this is found in the overlay budget as will be presented in the following section. Here the purely stochastic effects were accounted for by quadratic summation, whereas linear and weighted additions were used for systematic and mixed effects, respectively.

Application and user expectation. Another point of attention is the application of the product for which the budget is made. For example, assume there is a peak in the processor load, which occurs approximately once every hour and causes the product to slow down for a minute. Statistically seen, one would be inclined to omit this kind of deviation. However, whether it is important or not depends on the application in which the product is used and/or on the customers' expectations for it. For an office printer, for instance, the user will be annoyed for having to wait a little while, but will probably forget about it in the next half hour. The situation changes when the user expects the device to operate flawlessly, e.g. in case one is dealing with the design of a pacemaker.

5.3 Case study: The overlay budget for a wafer stepper

The budgeting method as presented in the previous section has been formulated in accordance with experiences gained in real engineering projects. The first case study considers the overlay budget of a wafer stepper, see also [82]. Wafer steppers are machines that produce integrated circuits out of slices bare silicon, so-called wafers. During the fabrication process, the wafers will be exposed to the patterns that need to be etched into them. This patterning takes place by a short wavelength light source in combination with an optical system and a pattern mask, called reticle. As a wafer is much larger than the area that can be exposed in one go, it needs to be moved (stepped) several times in order to expose the entire surface. This process is repeated many times, typically 16-25 times per wafer adding layers on top of previously exposed layers. To guarantee a correct alignment of the patterns, the movement needs to take place quite accurately. The accuracy is expressed as the amount of misalignment that occurs when a pattern is projected on the wafer. This so-called *overlay* is the topic of the budget. The *goals* of the overlay budget are:

- To summarize the requirements for subsystems and components.
- To get early feedback on the total overlay performance of a design, by being able to compare the results of individual component prototypes with the budgeted targets.

Scope of the budget

The budget was limited to those elements that directly influence overlay, such as the lens, the motion control system and the available sensors. The budget is valid under typical operating conditions.

Selecting a decomposition

The new overlay budget was based on the existing budget for a previous generation of equipment. Also the system decomposition into components had already been made before and could therefore be reused in the new budget. Note that this is a typical situation.

Finding quantitative figures

In first instance, the figures of merit for the relevant design choices were retrieved from measurement data that was already available from a previous version of the wafer stepper. This provided an initial budget. The existing budget for the previous version also comprised the contribution of each individual component to the budget total. Quadratic summation was used to account for stochastic effects, linear addition for systematic effects and weighted addition for mixed effects. These relations were based on an explicit model made by the system engineers. An example for quadratic summation is given by the *global alignment accuracy* in Figure 5.3. The figures for each of the three contributions do not add linearly (like it does for the *stage overlay*), but rather as a quadratic sum: $4^2 + 4^2 + 2^2 = 6^2$. At the same time, also a top-down approach was followed, since the new generation of wafer steppers needs a much better overlay specification than the older generation. The maximum allowable resource usage was set to a value determined by strategic road mapping [92]. Based on the road map, 80 nm became the required overlay target for this new generation of wafer steppers. This immediately pointed out the major design issue: all design choices needed to match this new resource requirement. This specification gave the upper bound and sufficient improvements were made with respect to the existing stepper to achieve this (as can be seen in the final budget in Figure 5.3).

Providing a clear overview

Figure 5.3 presents the overlay budget in a top-down decomposition. This overview must be read from left to right. The total allowable process overlay is 80 nm, which is split up into the contribution that each of the subsystems makes. After that, the contribution of all the main components is further broken down into different subparts and related to a specification for each part.

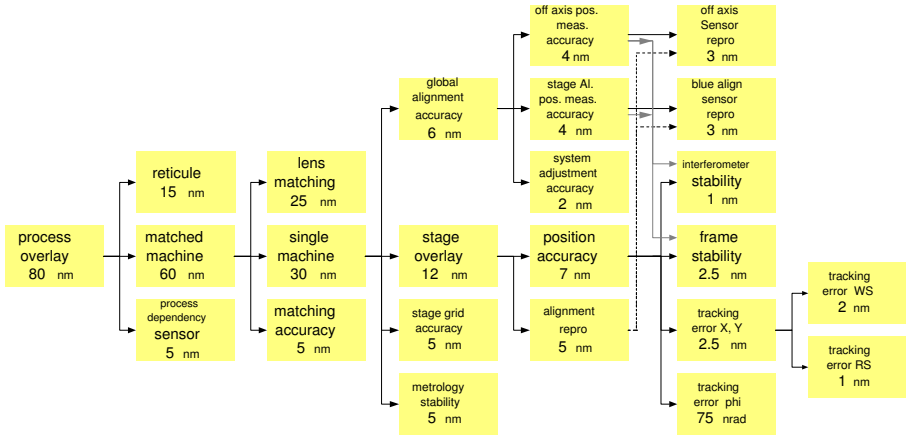


Figure 5.3: Waferstepper overlay budget

System adaptations

The initial budget had been based on an older generation of wafer steppers. Since the future generation had to have an improved performance, the initial budget lead to discussions among system engineers on how to satisfy the maximum overlay demand. In this interactive process, several design alternatives were discussed and some iterations on the budget were made. After a final round of negotiations the people involved agreed upon one of the solutions and they started converting the system-level budget into mono-disciplinary design decisions. This case study shows that a budget plays a crucial role in wafer stepper development.

5.4 Case study: The power budget for a copier

The project that was subject in this study dealt with the realization of a digital office copier, whose sales were targeted at a number of different countries. The latter condition implied that, in some of those countries, the copier needed to be operable under very strict power conditions. For instance, in the United States less than 2 kW is available from the power sockets found in an average office. As countries like the United States are important sales regions, the design of the copier was greatly influenced by the power issue. Being able to operate on the power from normal wall sockets became one of the most important realization aspects for this project. Moreover, other projects in the past had also struggled with power issues. As a result, power usage became a critical realization aspect, and was therefore subject of the budget.

5.4.1 Scope of the budget

Office equipment knows various modes of operation, like full production, low power or standby. Since a main driver was *'to be operable within the limits of a wall socket'*, this particular case focused on the situation in which the largest amount of power is consumed: during full production of the copier.

5.4.2 Selecting a decomposition

The next point of concern was to subdivide the layout of the copier into suitable components that influence power usage. Since a copier is naturally divided into a number of physical functions, this decomposition was largely maintained. An exception was made for the losses created in the voltage supplies and the cooling system. These losses could either be attributed to the functional components that cause them, or they could be grouped into the supply and cooling functions. During development, these losses are usually measured at the voltage supply side and at the cooling side. Therefore it was decided to group the losses into three blocks: the high voltage supply, the low voltage supply and the cooling system. On the other hand, if they had been attributed to each specific functional component separately, this would have explicitly shown the contribution of each component. The choice for a division in 3 groups was made for the sake of easy measurement and verification. The identified functional components are depicted in the graphical power decomposition, as given in Figure 5.4. The thickness of an arrow is proportional to the amount of power used by a component.

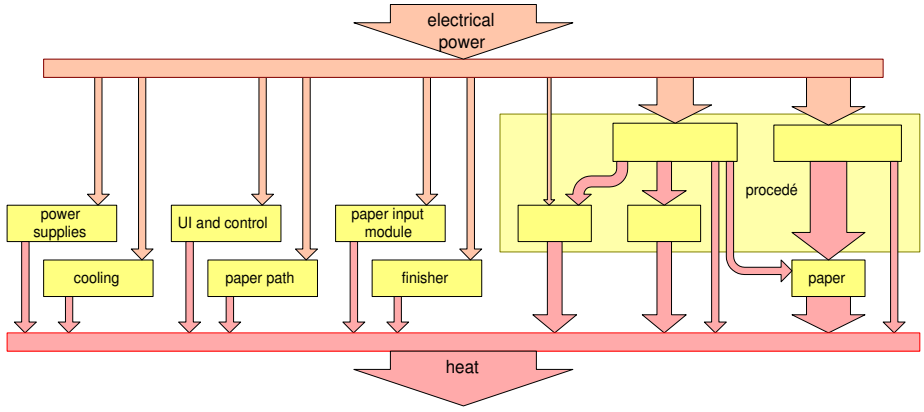


Figure 5.4: Graphical representation of the power budget of a copier

5.4.3 Finding quantitative figures

The largest part of the budgeting work was to find the realization choices in each of the functional blocks and to retrieve realistic values for them. The following paragraphs deal with some of the ways in which this was done.

Using a reference architecture in combination with third-party data. The *print engine control* unit was one of the components for which a reference architecture from a previous project was available. Although the exact configuration of its processing architecture was not known at the moment of the initial budget, the reference did prove to be a good indication of what power consuming components *print engine control* would ultimately contain. Once the components were known, data sheets from third party manufacturers of processors, memory and driver boards were used to make a reasonable estimate.

For those parts for which data sheets were not yet available, extrapolation on data of previous projects was carried out. Since manufacturers tend to give figures for predefined (worst-case) scenarios only, this approach only works well in case one is making a budget based on these scenarios. Once you need more specific values that depend on a non-typical mode of operation, hardly any concrete formulas are given. Finding the appropriate data may then become a cumbersome job. For example, how would one try to estimate the power usage of a processor when it has a 30% CPU load? Measurements on existing controllers or test bench prototypes are useful in these situations.

Using documented formulas. Another way of gathering data was employed in estimating the amount of energy that is transferred from the *transfer unit* towards each *sheet of paper*. In this case, internal documents contained the formulas for calculating the approximate amount of transferred heat. This kind of documentation actually forms an excellent basis for budgeting. Besides this information, (academic) literature often provides starting points for reliable budgets, too.

Using existing measurements. The power consumption of the *cleaner* and *preheater* units was estimated by means of data available from comparable units in existing products. Unfortunately, the copy speed of the new copier differed from that of its predecessor. An extrapolation on existing data was therefore performed. This introduced an extrapolation error whose margins were very difficult to determine. In order to reduce the associated uncertainty, measurements on a real test bench were done later on in the project.

Using prototype measurements. The amount of power absorbed by the *transfer* and the *cooling unit* cannot be expressed by formulas so easily. The estimates for their power consumption were based on measurements carried out on experimental setups and early prototypes. Unfortunately, this method leads to figures that are only valid momentarily and that - depending on their sensitivity to design changes - may become useless as soon as some details in the setup are changed. For this reason, figures found by means of experiments should be monitored throughout the project and validation in later stages might be necessary.

Simulation and modeling. The functional components that had the largest amount of uncertainty were the *paper path* and the *scanner* modules. Both of them are made

up of actuators that drive and control the flow of paper sheets through the copier and the scanner, respectively. Since paper transport is a time-dependent (dynamic) process rather than a static one, the figures for these components are the most difficult to extract. Of course, one could again take worst-case figures. However, since these worst-case figures only express power peaks that occur during accelerations, they provide a very conservative (too high) estimate for the overall consumption of the copier. That is, thanks to the spreading of the peaks, the overall power consumption will usually turn out to be lower than the sum of the peaks. On the other hand, once the peak power usage of the various actuators coincides, the resulting figure will become dramatically worse. For this reason, the time of occurrence (related to scheduling of paper flow) and the magnitude of these peaks must preferably be known prior to making predictions. In the studied case, this issue was analyzed by using a simulation model that was developed for scheduling the sheets in the paper path of the copier. Based on the velocity profiles for the actuators, the time-dependent power profile could be determined. The magnitude of the power peaks was retrieved from comparable actuators that were used in existing projects. So, based on assumptions concerning the paper schedule and the type of actuator used, we were able to reasonably predict actuator power consumption. Besides giving a sufficiently reliable indication of the actual power consumption, the resulting figures also made some of the project engineers aware of the fact that they needed to assure that actuators were driven such that none of the power peaks coincided. This is an example that shows how budget-based design positively influences a design process.

5.4.4 Providing a clear overview

After the information has been gathered, a budget needs to be presented to its stake holders. These stake holders could for instance be a project leader, a design team or even company management. In the copier case, the sales department might also be seen as stakeholder, since (small) power consumption can be a good sales argument. Since one of the goals of the budget is to communicate the effect of certain design decisions to others, its representation must provide a clear overview for all people involved in the project. For this reason, the tabular form might not always be the most ideal way to depict it. To really convince others by means of the budget, its data needs to be represented in a very informative way: preferably in a graphical depiction. Figure 5.4 presented the functional decomposition of the copier, together with the collected data on power consumption. The *power decomposition* in Figure 5.4 should be read from top to bottom. The upper horizontal bar represents the source of the system resource that was budgeted (the available power, less than 2000 Watt), whereas the lower horizontal bar expresses its sink (the dissipated power). The blocks in between the source and the sink represent the functional modules that consume the resource. Recall that these are the same modules as identified during step 2 of the budgeting process. The arrows in between source and sink express the resource flow (power flow) from source to modules, mutually between modules, and from modules to sink. The arrows have been

scaled to show their relative influence on the total resource flow. For what the studied case concerns, this tells you in the glimpse of an eye that the largest amount of power is consumed in the modules at the right side of the figure: in the *transfer* and the *preheating* units. Such a conclusion gives system architects a convincing argument to demand for considerable improvements in the responsible modules. So, although Figure 5.4 expresses the same figures as could have been written down in a simple table, it does give a much better overview on the whole situation. Moreover, for quantitative details a table can be consulted anyway. According to our experiences, visual depictions have proven to be much more informative in the sense that people immediately understand how the resource is distributed over the various components in the design. Hence the visualization is a major factor in creating insight in the distribution of a resource.

5.5 Budget dynamics

Resource budgets applied in engineering environments can be categorized roughly into two types:

Static budgets. The word *static* signifies that budget values do not change with the particular operating mode of a device. Typical examples are average or worst-case values. Note that static figures do change when a design changes after some time.

Semi-static budgets. A *semi-static* budget means that several static budgets are made: each for a different mode of operation of the product. The budget for a copier, for instance, can have a version for full production as well as one for standby of the device. Semi-static budgets are often used in combination with scenarios. For each available scenario or use case (see e.g. [33]), a dedicated budget is made. Each budget then solely contains the effects that occur under that particular condition.

If the resource cannot be properly captured by static or semi-static budgets, often (simulation) models are used. Typically, when the dynamic time-varying behavior of components play a major role in the budget, it is best to use simulation models and tools instead. We already discussed the example of the paper path in Section 5.4.3 for which the worst-case power usage (the power peak) of the paper path could not be derived from the power peaks of each individual motor. The reason was that the individual power peaks are spread out over time. As a consequence, the total peak power depends heavily of the distribution over time of these individual peaks. This is prohibitive for making a useful decomposition of the paper path into smaller subcomponents for the purpose of a (static or semi-static) power budget for the copier. The paper path should be taken as one entity in the decomposition. In this case the use of a simulation model for this part of the system is advised. An example of a simulation run is given in Figure 5.5. This plot depicts the sum of the power usage of some motors in the paper

path of a copier, over time, during a copy job. The outcome of a simulation model can be used to find the corresponding figures that are required in a (semi-)static budget.

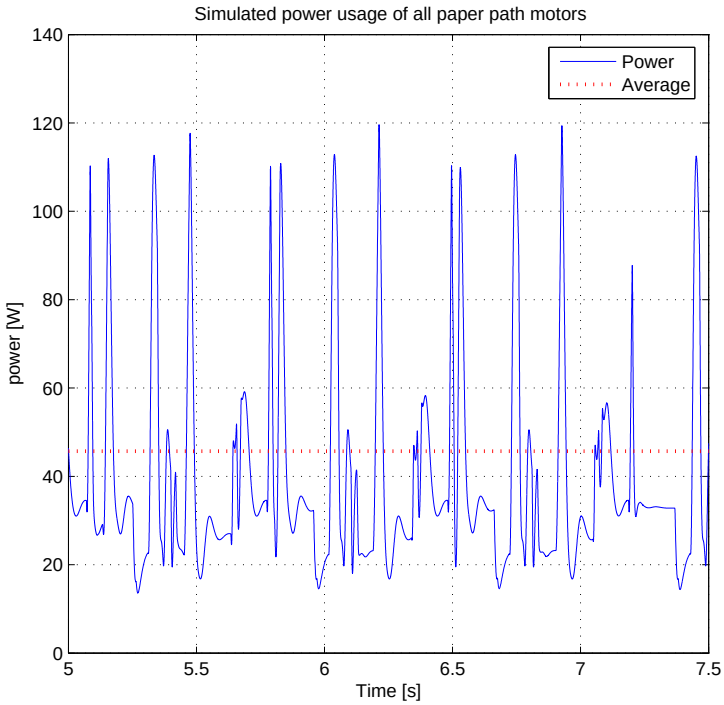


Figure 5.5: Simulated motor power usage

5.6 Budgeting benefits & concerns

The case studies presented in this chapter show how the proposed budgeting approach can be applied in a systematic way. The outcome of these studies was that consistently applying budget-based design techniques has various advantages. These advantages are described in the following paragraphs.

5.6.1 Benefits

Identify threats & (early) design improvements. One of the merits of making budgets is that budgets clearly document the particular specification of a project. They state the available resource and the way in which it is supposed to be distributed across the

design. In order to compose a budget, project engineers first have to think thoroughly about a design for the system, the decomposition and how to quantify the components with realistic values. Since different engineers have different opinions, this process will undoubtedly lead to discussions and negotiations, based on facts rather than on feelings. Discussing a design in order to create commitment between engineers often reveals a number of critical realization aspects. Such a priori knowledge of potential problems enables one to adapt the design on time. Since budgeting is often started in the initial stage of a design, optimizations can still be made before any irreversible decisions are taken or costly mistakes are made.

To make the design explicit for communication and awareness. A side effect of discussing the budget is that the exact meaning of the quantities in the budget gets documented and communicated quite effectively towards all project members. Once implicit design assumptions are made more concrete and are talked over by engineers, they tend to develop a larger awareness of potential design risks. In this way everyone will become more aware of the consequences of the decisions they take and sees how this fits in the bigger, multidisciplinary picture. In this sense the budget provides a baseline to take decisions.

Enabling concurrent design via specification of requirements for components. Besides contributing to design improvements, budgets also enable concurrent design of parts for which the specification can be extracted from the budget. Once the budget has been written down, it expresses the exact distribution of the resource over the individual functional modules. The budget itself is used to ensure that the interaction between the individual components satisfies the overall resource requirements for the product. This gives also clear guidance during the integration process. Moreover, since a budget records the agreements that were made, people will also tend to argue less about them afterwards. Potential conflicts can thus be avoided, which is good for the working atmosphere.

Managing design margins. Although this was not emphasized in the particular case studies, one of the large advantages of budgets is that they enable making design margins explicit. On the one hand, budgets can include the uncertainties due to unknown effects and missing information, whereas on the other hand, they can also include the design margins that can be obtained by improving the design (at the cost of longer design time, larger design effort or higher cost price). By making uncertainties more concrete, risks can be identified and reduced in an early stage. If improvements are required and the budget offers various options to achieve them, a suitable choice can be made.

Support road mapping. As we saw in the case study of the overlay, budgets can also serve as a tool to road map future versions of one's design [92]. The future evolution of the budgeted resource can be strategically relevant to the company's success. Budgets indicate the strengths of the current design and can therefore help to identify the necessary technological steps to achieve the improvements to be realized.

5.6.2 Concerns

One of the major concerns for using the budgeting method in a project is that one needs to watch out for that people 'hide' themselves behind the figures in the budget and that become inflexible to changes in the figures, in a later stage. In early stages of the design the figures might still be uncertain and subject to change. People have to realize this and appreciate the value of the budget anyway, since it makes the design issues explicit. People that use the figures in the budget as an excuse for not having a proactive attitude, no longer take their own responsibility for the quality of the overall product. This phenomenon requires an attitude change for some persons, or sometimes even a cultural change within a company.

5.7 Conclusions

In this chapter the system-level method of budget-based design is promoted. An iterative approach consisting of four steps was given to set up a budget. In addition to the iterative approach, guidelines are given on how to retrieve the relevant data and how to deal with uncertainties and design margins. The proposed budgeting approach was applied in two practical cases. During these cases, several advantages were recognized. Budget-based design supports:

- Early recognition of potential threats to meeting the requirements for the budgeted resource. This provides guidance in taking the correct design decisions and in making improvements.
- Making the design more explicit and transparent. This enables improved communication of project targets, leading to a greater awareness of the consequences of individual decisions on the design as a whole.
- Concurrent design and integration. Proper documentation of the resource distribution in a design can be seen as a specification for the individual components. This enables an improved concurrent design process and provides a baseline for integration.
- Managing design margins explicitly.
- Identification of the technological leap to be taken in future, therefore assisting the road mapping process that is carried out with the goal of keeping the business successful.

Taking all pros and cons into account, we believe that its systematic approach supports making decisions that reach over the mono-disciplinary engineering disciplines and over the functional modules of a product. Thereby it is a multidisciplinary design method that supports product design. This chapter can be used as first start to learn the skill of budgeting.

Chapter 6

Effective industrial modeling: The example of Happy Flow

Authors: J.M.J. Beckers, W.P.M.H. Heemels, B.H.M. Bukkems and G.J. Muller

6.1 Introduction

In various branches of engineering, modeling plays a central role. As such, it finds also its place in the design of high-tech systems like copiers, wafer steppers and televisions. In the design of these high-tech systems multiple disciplines need to make the overall design in close co-operation. For instance, the electronic design, mechanical design and software design together need to describe a consistent, functioning machine. The designs are often made in parallel by multiple groups of people, where the communication between these groups is hampered by lack of common understanding. In addition, the complexity of a copier (typically millions of lines of code, thousands of mechanical components like frames, springs, and belts, and many motors, sensors, and printed circuit boards) give rise to many cross-disciplinary design decisions. To make a good tradeoff, the overall effect of a design decision needs to be evaluated as early as possible. This is where models come into play. On one hand models can be used to predict and evaluate the effect of possible design choices, even when the machine itself has not been built yet. In this stage models support taking design decisions. On the other hand, models can capture design decisions and can create a common understanding that bridges the gap between the disciplines involved in the design. However, even when using models, physical prototypes are essential because of the confrontation with physical reality, where overlooked issues will inevitably pop up.

Models appear in all kinds of forms; they range from simple drawings or sketches of the layout on blackboards to detailed models (e.g. differential equations for describing

physical processes or finite state machines or automata for computer programs). In this chapter we are interested in the question which properties a model should have to be effective from an industrial point of view.

As already seen in Chapter 1, the Boderc goal is to develop a design methodology based on multi-disciplinary modeling to predict the performance of a system in the early design phases (see Figure 1.2). The aim of the methodology is to reduce the overall design effort and time and is based on the philosophy of shorter cycle times between design phases. The latter is expected to be achieved by models that can be built relatively quickly and generate reasonably accurate predictions of system behavior. To stress this point, very accurate modeling is sacrificed to reach a fast iteration through various model instantiations. Already the use of models has the advantage that they enable a much faster evaluation of different design options if compared to physical prototypes. The reason is that a new prototype would needed to be built for each design option, which is very time consuming. Through analysis of models different designs can be evaluated much faster.

Several models have been proposed in the Boderc project to support the design of a copier, as evidenced by this book. Some of these models were easily used by the industrial partners, while others did not find employment. This indicates that there are specific properties that make a model a success in industry. To identify why certain models are embraced by industry so easily, we consider in this chapter the most successful industrial model created within Boderc. This model focuses on the design of the sheet transportation system in a copier. By identifying the success factors of the model, we aim to indicate how modeling can be improved from a point of view of industrial usefulness.

6.2 The design problem

The copier context has already been described in Chapter 1. The focus in this chapter is on two levels of the copier design, although they heavily interact and influence each other:

- the layout of the paper path,
- the scheduling of sheets.

In Figure 6.1 a more detailed drawing of a paper path is given.

6.2.1 The layout of the paper path

Several issues play a role in the design of the transportation system of the sheets in the copier. The model of the transportation system consists of several parameters of which the *layout parameters of the track* are the first that come to mind. Next to this layout, the *drives* of the sheets have to be selected, i.e. the *pinches* and *switches/flips* to direct

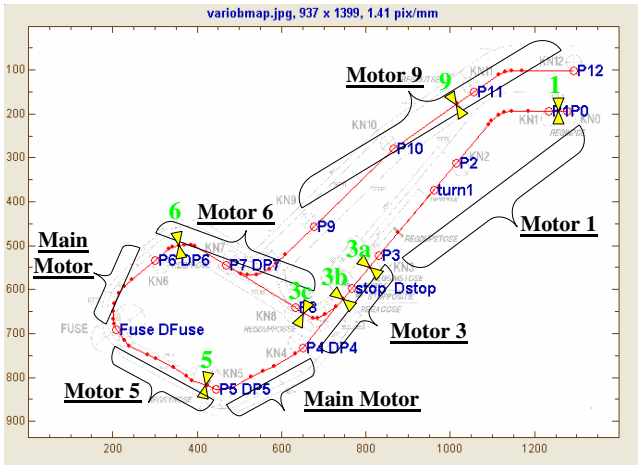


Figure 6.1: Paper path, with positions of the pinches, bypass and duplex loop.

the sheets into the right track. The pinches and switches require *actuators* like *motors*. Moreover, *sensors* have to be present to detect the presence of the sheets.

The layout of the track has to be such that some functionality of a copier is guaranteed:

- A *turn* loop has to be present to enable duplex, i.e. two-sided printing,
- *Registration and synchronization* are necessary to accurately adjust the sheet position in accordance with the images,
- The *fuse* or copy press is the location where the images are printed onto the sheets,
- A *heater* has to be present in the track such that the temperature of the sheets is increased to a desirable level for the fusing process. The track has to provide space for this.
- *Start and destination* of the track. The sheets have to be inserted from the paper trays at some point and have to leave the track again at the finisher. Sometimes the paper input module, the fuse and the finisher are at fixed locations due to standardization.

Within these 'functional' constraints, in principle everything is possible. Although several other constraints, some based on specifications of the copier and others based on previous design experience, apply to the track. The size of the copier forms one of the most severe restrictions on the paper path, but also the maximal curvature of a curve is constrained due to bending properties of heavy sheets of paper.

6.2.2 The scheduling of print jobs

Given the layout of the track, it still has to be determined how the sheets will move through the paper path in the sense that the position and velocity profiles over time have to be determined for individual sheets (called the timing table), but also for complete print jobs. In the scheduling of a print job, the sheets motions have to be coordinated with respect to each other and for instance collisions between sheets have to be prevented. The print schedule has major implications for the total timing of the engine as most other actions in the copier are synchronized to the schedule. Indeed, from this schedule one derives the motor profiles, the requirements on motor characteristics and control algorithms, the sensor triggering and the real-time response properties of the software, the timing of the imaging process and its related subsystems and so on. Hence, the scheduling has a large impact on the total success of the copier. The scheduling is of course depending on how the mechanical layout is chosen and actually this layout imposes constraints for the schedule. For instance, if there is a certain time needed to open a closed switch/flip, this indicates that certain margins between two sheets that have to take different routes at the switch must be included in the schedule. If a desirable scheduling of the sheet flow cannot be realized guaranteeing for example a certain throughput of the machine, then an adaptation of the mechanical layout is necessary.

6.2.3 Requirements

Various key drivers and system requirements should be satisfied when designing a new system. Key drivers from the customer's perspective are for instance minimal waiting time, ease-of-use and (re)production quality. These key drivers translate into various technical system requirements like throughput (pages per minute), position accuracy of sheets, time-to-first-print, et cetera. See Chapter 3 for an overview on the key driver model for a particular printer. Of course, also many other (resource) constraints like power usage, cost price, size, et cetera, play an important role. The choice of the layout of the track and the scheduling have a major influence on several of these requirements:

Energy and power usage: these are related to acceleration, velocity and forces required for the transportation. Energy and power usage have strict constraints; objectives related to energy labels like Energy Star play an important role, whereas maximal power usage is directly coupled to the maximal power available from a normal wall socket in the country of interest.

Low costs: by using a simple concept of control, cheap and few actuators by combining drives (e.g. one motor controlling multiple pinches), the cost price of the system can be kept low.

Throughput and time-to-first print: realize that certain print jobs are finished quickly.

Synchronization, printing accuracy and registration: make sure that a certain reproduction quality is obtained. This requires tight synchronization and positioning of sheets and images.

Low complexity of control concepts: keep the development and the size of the control software manageable.

Size: the size of the resulting copier.

In order to facilitate the design of the mechanical lay-out and the scheduling, a model called *Happy Flow* was developed within Boderc.

6.3 Model-based design: Happy Flow

The name Happy Flow is based on the conscious simplification of the model, where only the desired behavior of a sheet and the ideal movements of all parts are modeled. All disturbances and variations of actual hardware performance are ignored. It is a kinematical model, where non-idealities such as friction and limited jerk (derivative of acceleration) of motors are not taken into account.

The main goal of the Happy Flow model is to perform a quick design space exploration with respect to the job scheduling. From the insight obtained in this phase also the mechanical layout can be adapted in a cyclic design procedure (cf. Section 6.4 below). For the fast design space exploration the following subgoals can be distinguished:

- Easy specification of a 'happy flow' schedule of print jobs.
- Fast checking if for a certain happy flow, for a given paper path, for all required sheet sizes and operation modes, the design requirements like throughput and power usage are met and the constraints (safety distance between time, sufficient time to put switches in right position, et cetera) are not violated.
- Verification of the robustness of the happy flow for implementation.
- Demonstration and inspection of the details of the happy flow in an easy manner, such that it can be optimized manually.
- Generation of timing tables, speed settings, and the expected times of arrival at specific points, that will be used in the software that controls the paper path.

The prerequisite that is needed to setup the Happy Flow model is an (initial) mechanical layout of the paper path including the position of pinches, switches, et cetera. As already mentioned before, Happy Flow is a high-level (idealized) model where all sorts of low-level (non-ideal) effects are ignored. However, the most important effects are taken into account in the model, for example critical software delays, actuation delays (e.g. solenoid delays for setting a flip), and a maximum acceleration and deceleration rate are incorporated.

6.3.1 Basic working of Happy Flow

The Happy Flow model started as a small and simple simulation, where logistics and timing information was combined to generate position-time diagrams for sheets in print jobs. The availability of the input data for this simulation was also convenient for generating an animation, superimposed on a drawing of the paper path. A next step was to generate the input data for the Happy Flow model directly from available mono-disciplinary design data, such as CAD drawings. This evolution is shown in Figure 6.2.

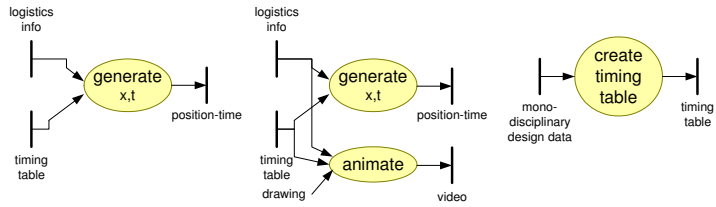


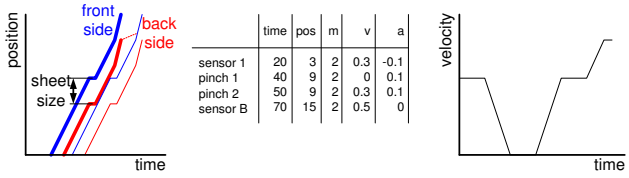
Figure 6.2: Incremental growth of the happy flow model. From left to right: initial form, intermediate form and latest addition.

The current version of Happy Flow starts from the CAD construction drawing of the mechanical layout. Important registration points at the CAD drawing are indicated which are necessary to capture the mechanical layout in a computerized two-dimensional format. Also important points at the layout (like pinches, sensors, switches, et cetera) are included as registration points. Typically the computerized two-dimensional information of the track consists of the coordinates of registration points and an indication how the track between them is connected, which is typically via linear interpolation. Together with individual sheet info (e.g. the length of the sheet, duplex/simplex printing, its source and destination, et cetera), a 1D track is constructed. This 1D track is the one-dimensional view on the track the sheet has to travel. It consists of a concatenation of all the registration points the sheet has to pass, together with the total traveling distance. The 1D track information together with certain hardware parameters will be converted into a timing table (see Figure 6.3 for an individual sheet that can have several representation forms: position-time diagram, timing table and velocity profile). Hardware parameters are for instance the relative velocity of specific pinches with respect to the fuse speed, necessary stopping times at certain positions to perform specific actions, et cetera. The information like the number of sheets in a job, the ordering of sheets in finisher, et cetera are collected in job info and converted into a schedule. The job schedule typically consists of the starting times of each individual sheet in the job; as each sheet of the same format will follow the same position-time diagram, the starting times are sufficient to specify a print job. For the animation typically a jpeg or bitmap picture of the mechanical layout (typically a simplification of the CAD drawing) is used to display the motion of the sheets through the machine. This animation is interactive and can run forward or backward at any speed. It can also animate motors

and switches when they have their own specific ‘happy flow’ specified. The animation of sheets, motors, and switches, gives good evidence that the model can work in reality. This certainly helps to avoid many unpleasant surprises afterwards.

6.3.2 The scheduling of sheets in a job

The basis of job scheduling is that the design assumes that each sheet of the same format always follows the same position-time diagram. The job schedule is typically based on the desired throughput (pages per minute) that has to be achieved. This throughput, the sheet format and desirable inter-sheet distance determine how fast the fuse (where the actual printing process takes place) should run. This creates a fixed rhythm of the copier in steady state operation, which can be translated to the starting times of the individual sheets in the Paper Input Module.



	time	pos	m	v	a
sensor 1	20	3	2	0.3	-0.1
pinch 1	40	9	2	0	0.1
pinch 2	50	9	2	0.3	0.1
sensor B	70	15	2	0.5	0

Figure 6.3: Multiple representations used in the model: position-time diagram, timing table and velocity profile

6.3.3 Animation

In the animation the timing tables are used and the 1D track developed in the model structure, i.e. the track being described on the position axis of the position/time diagrams, is mapped back to 2D and one can see the actual movement of the sheets in the print job through the paper path. As a background the CAD drawing (or a derivative of it) is used for this purpose. The animation is interactive as you can run it on any percentage of the real engine speed. Sliders and keys can be used to slow down or speed up, and you can step forward or backward to any situation, take snapshots (‘photos’) to illustrate documentation and make movies for presentations. Figure 6.4 gives an impression of what this looks like. In the upper left corner an active table of all the sheets that are currently being transported can be seen. The time until they reach the fuse for the first time (first column) and their actual position (fourth column), velocity (fifth column) and acceleration (seventh column) are given. One can imagine that this provides a very insightful means to visually inspect all kinds of behavior in the paper path.

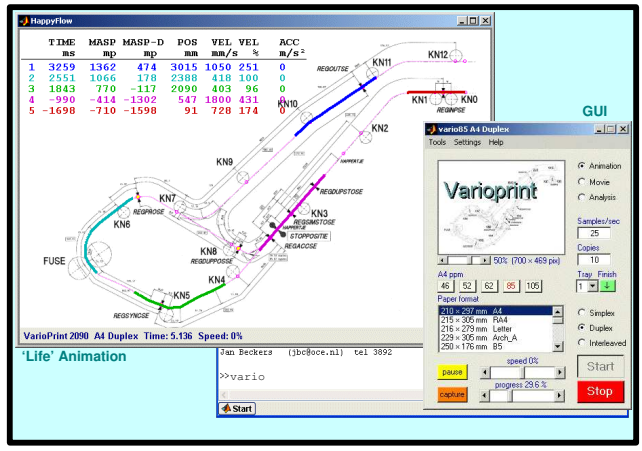


Figure 6.4: Impression of the Happy Flow animation

6.4 Design cycle

Typically, one designs the timing tables for the common sheets sizes A4 and A3 and for simplex and duplex and derives via minor adaptations the timing tables for other sheet sizes. If the problems cannot be solved at the level of the scheduling (timing table and job scheduling), changes in the mechanical layout might be necessary. Several iterations might be necessary in the design cycle as depicted below.

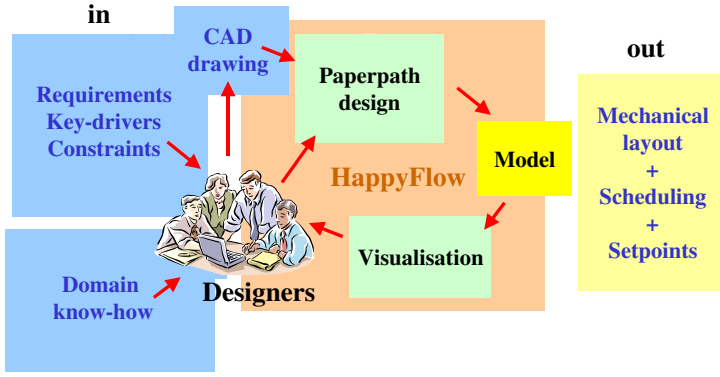


Figure 6.5: The Happy Flow design cycle

6.5 Industrial success factors and conclusions

Besides describing the basic working of the Happy Flow, the original goal of this chapter is to identify industrial success factors of modeling. Based on the success of this particular model, we will provide a list of the reasons why this model was so easily introduced in and used by industry. As indicated in the introduction, the goal of the Boderc design methodology is to enable fast model-based design space exploration in the early design space by predicting system performance. The main drivers for this Boderc goal are the business objectives time-to-market and *design effort/cost* while keeping *industrial constraints* like *maintainability* of a model and *human resource constraints* in mind. By human resource constraints we mean that the model should support the engineering approach used in industries. Typically this requires that the model should be *easy to learn* (low initial investment of time and effort to learn to use the model in an *effective manner*) and *easy to use*, properties that various academic models often lack. For short time-to-market a *short cycle time* of the application of the model is needed. This means that the model should be easy to build and should have a reasonably accurate predictive power. The right balance between accuracy and design time is important. The business objectives time-to-market and low design effort/cost are realized by five sub-drivers being: using a model instead of a prototype, short calculation time for the model, stimulate cross-disciplinary communications, approach the right problem (which is crucial for the overall system design) and find relevant information in the model (and of the to-be-built system) easily. In Figure 6.6 we represented the above reasoning graphically. All the above mentioned drivers that realize the Boderc goal are in the end related to 12 issues that played from our perspective a role in Happy Flow's industrial success. These 12 issues are:

A. **Modular setup.** Happy Flow has a modular setup. The complete model and program consists of smaller parts that are connected through input-output relations (see also Figure 6.2). By suitable concatenation of these subprograms one obtains a high-level function that can be easily interpreted. This is important for understanding, insight and maintainability of the tool.

B. **Stepwise introduction and feedback.** Little steps in the evolution of the Happy Flow model made evaluation towards industrial practice possible and of course, the success of the individual steps led to a stepwise introduction at the copier manufacturer (see also Figure 6.2). Moreover, this also enables that feedback was given during the development of Happy Flow, which lead to frequent refactoring of the tool to keep its structure useful and practical for its users and purposes.

C. **Limited size.** The size of the model is limited (one thousand lines of code). This is important for understanding, insight and maintainability of the tool. The model size also has effect on speed of execution.

D. **Use of conventional paradigms.** The conventional representations of timing tables and position-time diagrams are still present in the model or can easily be

generated. Hence, the outcomes of the model can still be easily communicated and transferred to all people, which are familiar with timing tables.

E. **Right representation at right place.** Several variations are used that represent the timing table for the motion on an individual sheet (see Figure 6.3). The representations can be converted into each other, so that for the particular purpose the 'best' representation can be selected and easily generated. 'The right man at the right place' so to say. This has a positive effect on speed of computation as well.

F. **Good level of abstraction.** The model has a good level of abstraction. It is not too detailed. The distance to system level key drivers like throughput, power usage, size, et cetera. is not too large so that it helps to make system level trade-offs. The model is not too coarse either as it still predicts the basic timing of the sheets within reasonable accuracy. The Happy Flow model is directly connected to the design of subparts of the machines like selection of motors, real-time software, et cetera. Hence, on one hand it assists in predicting important system level drivers like throughput, power usage, et cetera, but on the other hand it also couples to mono-disciplinary design problems.

G. **Simple and fast computations.** The computations that have to be performed are very simple. This enables fast calculation of the model and thus gives answers in short time.

H. **Conceptually simple.** Happy Flow is conceptually easy to understand and as such can be used for reasoning and communication across disciplines. This supports breaking down the communication barriers, which are often present in multi-disciplinary designs.

I. **Addresses right design problem.** The model addresses an actual and current design problem. Although engineers could solve it in the past by large investments of time and effort, the introduction of Happy Flow was able to gain much in design time. It was a latent design question. Outcomes of the model are important for the overall design in the form of event or signal tables.

J. **Easy visual inspection via animation.** Visualization and animation on a picture of the mechanical layout (CAD drawing) makes it easy to interpret the results and makes them insightful.

K. **Data base.** In the use of Happy Flow one continuously makes assumptions for design issues that are currently unknown or not documented. By doing so one stimulates discussion and modifications in the assumptions by showing the effects via visualization. Hence, consensus is created for these assumptions and this implicit domain knowledge is somehow 'documented' in the Happy Flow model. In this sense Happy Flow also has the role of a database with the latest design specs.

L. **Easy validation to reality.** It is easy to compare model output to reality (validation of model). Significant differences can be adjusted. Little deviations can be attributed to the good weather conditions under which Happy Flow works.

A 13th issue is indicated as 'Prediction of system performance,' which goes almost without saying as this is what a model within the Boderc context should typically do (see Figure 1.2).

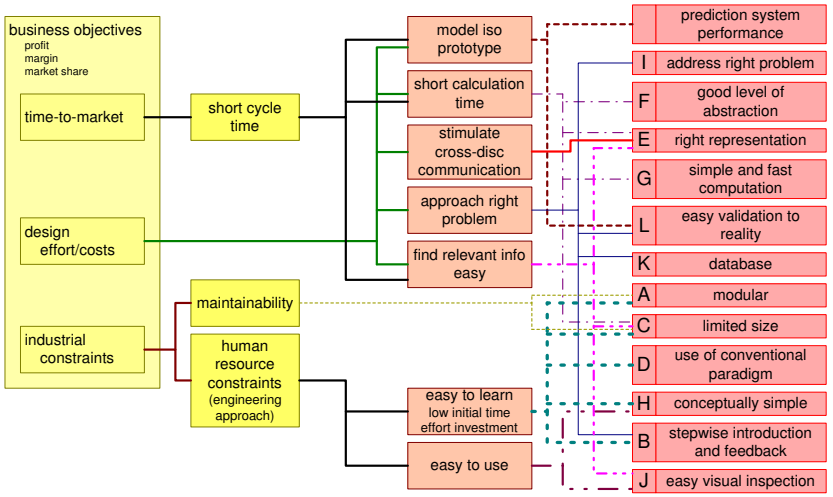


Figure 6.6: Overview of success factors of Happy Flow

All these factors contributed to the fact that the Happy Flow method is now used by industry and the results are promising. Engineers embrace it, explore the design space in shorter time, extract all kinds of information from it, use it for measurements and use advanced spin-off models. The designers have confidence in the model and drastic changes in the mechanical layout are now easily handled without hesitation even in critical phases of the development. Moreover, the good prediction capabilities of Happy Flow and the overview of the total paper transportation system that it provides, enable less conservative designs. As risks and uncertainties are reduced, certain (more optimal) designs are admissible at present, while these would be discarded in the past. From an even broader, system engineering perspective, it is important to learn from such instances of successful industrial models. The identification of the success factors is a first step towards a more systematic method that gives clear guidelines on how to create industrial effective models that support the system architects and speed up the multi-disciplinary design of high-tech machines. This chapter forms a first step as only one successful model is considered, but future work of the Embedded Systems Institute focuses on finding such a method.

Chapter 7

Heat modeling in copiers

Authors: E.H. van de Waal, J.M.J. Beckers and J.F. Broenink

7.1 Introduction

Heat is one of those system aspects that is influenced by multiple engineering disciplines. Because it is difficult to communicate design decisions between disciplines, as discussed briefly in Chapter 1, it is difficult to find the optimal design with respect to these system aspects. One way in which the design process can be improved is to use model-based design. Using models, it becomes possible for the disciplines to communicate insights and make trade-offs in an early stage of development, before design changes become prohibitively expensive.

In this chapter, an example of a heat model is shown that can be used for these purposes. First, the relevance of heat in a printer is explained by linking heat to the fundamental goal of a printer: making high-quality prints. This gives insight in the questions to be answered by modeling. Then some modeling techniques for heat are discussed, and a case is studied in which a heat model is constructed.

7.1.1 Relation to key drivers

Heat can lead to many potential design issues. To streamline the design process, priorities need to be assigned to potential issues. To assign priorities, heat should be linked to the key-drivers (see Chapter 3) for a printer. Thus, the effect of heat flow on the customer perception of the printer can be investigated.

A customer can observe the following effects of the heat house holding:

- The amount of energy needed to establish the temperatures required by the printing process determines the amount of time needed for the printer to warm up in preparing for the first print (*Time-to-first-print*).

- The temperatures of paper and toner at the moment they meet in the fuse pinch, strongly define how much and how deep the melted toner penetrates into the paper fibres and strongly influences the printing quality (one of the key drivers for an Océ printer). As such, this effect is also taken into account in the printing accuracy model of Chapter 9.

- The amount of heat lost during printing affects the speed at which the printer can print. Throughput is also a key driver, as shown in Chapter 3.

- The average energy consumption of the printer over a normal 'office' day is part of the running costs for a printer. Most power consumed by a printer is used to maintain heat flows. Also, power usage has been restricted by environmental agencies.

- The peak energy consumption of the printer partially determines how easy it is to install a printer at a customer site. This is nation dependent: in Europe, a regular socket can supply roughly twice the power a regular socket in the USA or Asia can supply.

These issues serve to illustrate the importance of proper heat management for a printer.

7.1.2 Temperature constraints within a printer

Printing is a physical process in which a toner image is produced using electro-magnetic forces, which is then transferred to the paper and fused. Fusing is performed by applying heat and pressure to the toner and paper, such that the toner melts and enters the paper. In Figure 7.1 the main parts of the VarioPrinter 2090 are shown: the pre-heater (VVW) that rises the temperature of the paper before fusing; the imaging unit that generates the toner image; the Toner TransFer belt (TTF) that tansports the toner from the imaging unit to the fuse pinch, and the fuse pinch where toner is fused on the paper. While intended to increase the life-time of the imaging unit, the TTF also separates the warm and cold areas. The printing process is described further in Chapter 1.

Most of the energy consumed by a printer is used to establish and maintain these temperature levels, and many key drivers are influenced by the resulting flow of heat. As an example, the key driver *Time to first print* will be taken. This time must be as short as possible, and is directly determined by the speed with which the temperature levels can rise to acceptable levels in a printer. Whether the temperatures are 'acceptable' is determined by the following constraints:

- If the fuse temperature is too low, the toner will not penetrate the paper properly. It will form a layer on top of the paper which is easily removed through bending and scratching.

- If the fuse temperature is too high, toner will be too fluid and may be smudged by mechanical contacts.

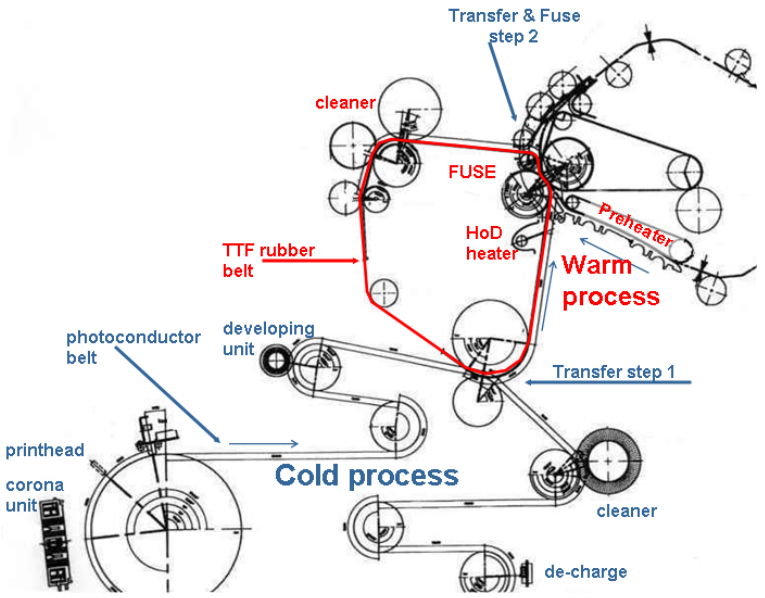


Figure 7.1: Printer process overview

- When forming the toner image, toner must not stick to any surfaces except to the paper where it is intended. This means that the temperature of those surfaces must stay well below the fusing temperature to prevent accidental fusing and soiling of these surfaces.
- Toner and paper dust will invariably contaminate parts of the rubber belt with which toner is fused to the paper. The surface is cleaned using a special drum that is sufficiently warmer than the surface being cleaned. In this manner toner sticks to the cleaning drum and traps the paper dust.
- Temperature constraints apply to the imaging unit to prevent damage.

A printer will usually not produce prints continuously. Often, the printer will enter stand-by mode when no jobs are being processed. During stand-by the temperature levels must drop as little as possible to minimize the time to first print. If the printer is not used for a longer period, it should enter sleep-mode and finally the power should be switched off, to meet environmental requirements. During these phases, the temperatures must also drop as little as possible. The cool rate is determined by the construction of the printer. This leads to a number of trade-offs:

- The heat capacity of the printer should be large during printing and stand-by, and small during warm-up.

- To minimize the heat lost during stand-by, the printer should be thermally insulated. However, while heat is transferred from hot areas to cool areas during cooling, it must be ensured by the design that the maximum temperatures for the cool areas are not exceeded.
- The rollers should have high thermal resistance to minimize heat loss, but should have low thermal resistance to equalize heat distribution e.g. over the rubber TTF belt.

Model-based design can be used to meet the constraints and find the optimal trade-offs.

7.1.3 Heat modeling goals

If a model is to be used in model-based design, it must give answers to specific questions. For example, an important printer property is the time required to warm the printer when it has cooled after being shut-down for a long period. Environmental agencies specify the maximum warm-up time. It is one of the main figures of merit for a printer. An interesting question is if a current design will have an acceptable warm-up time. A model can be used to give the answer.

Various strategies can be used to heat the printer, by distributing the available power in different ways to the various parts of the printer. Using a prototype, different methods of heating the printer can be tested empirically. However, only one such experiments can be performed per day, due to the need for the well insulated printer to cool before the experiment can be repeated to test an alternative strategy. Thus development time can be gained if a suitable model is available to determine which strategy is optimal, and how the temperatures react when a printer starts printing after warm-up.

Another reason to use modeling is that it gives insight in how the current design can be optimized. For example, a model gives insight into which are the dominant heat losses, so that efforts to reduce loss can be focused on the areas with the most impact.

7.2 Heat modeling techniques

Heat is a property of matter. Each chunk of matter stores heat, and achieves a certain temperature depending on the amount of heat stored. This behavior can be modeled similarly to the storage of electrical charge. For example, a capacitor stores electrical charge, resulting in an electrical potential over the capacitor. Another analogy that can be used is the storing of fluid in a container, which results in a pressure at the bottom of the container. Here, the electrical analogy will be used most often. Just like the storage of heat can be modeled similar to the modeling of charge in a capacitor, the transportation of heat can be modeled similar to electrical charge, which is transported through resistors. As with electrical resistors, heat resistors are assumed to *not* store heat.

Heat can be transported through conduction by the mechanical construction, convected through the flow of air inside the printer, and radiated by hot surfaces. Also,

heat is transported by the movement of objects (e.g. sheets or the TTF belt) through a printer. Heat conduction and convection are linear resistance effects, although the resistance may depend on system parameters e.g. rotation speed. Heat radiation is a non-linear resistance effect, following the Stefan-Boltzmann radiation law [16].

Heat is inserted into the printer as electrical power. Most of the electrical energy consumed by the printer is turned into heat. A printer has several heaters which generate heat from electrical energy. They can be modeled as perfect heat sources.

The heaters, the mechanical contacts through which heat is conducted, and the air flows inside the printer that are responsible for convection, are often controllable e.g. through actuators. As such the heat resistances must be modeled as variables, and their control is to be included in the model.

Using heat sources, heat capacitors and heat resistances, thermodynamical models can be made. Two main methods exist: finite-element modeling where mechanical parts are modeled in detail, and lumped models where the thermal properties of parts are lumped together.

7.2.1 Finite Elements Models

In Finite Elements Models (FEM), the assumption is that real matter consists of an infinite number of infinitely small heat capacitances interconnected by heat resistors, and that reality can be closely approximated using a fine-grained grid of partial differential equations. This can be done in one, two or three dimensions, depending on the modeling needs.

FEMs are well suited to determine the temperature distribution of an object or system of objects, and to predict the actual thermal characteristics of these objects. However, when studying a complete printer, this modeling technique is too detailed: to keep model complexity acceptable, only the average temperatures of objects and average object characteristics can be taken into account. Océ uses FEMs to optimize the shape of individual components in a printer that are related to heat distribution, such as, for instance, a pre-heater.

7.2.2 Lumped heat modeling using bond graphs

In a lumped model of a printer, the model is simplified such that only the most interesting temperatures are explicitly included. The assumption is then that there are a number of parts that directly determine this temperature, and that these can be lumped together into a single heat capacitance. The interaction between areas of different temperatures are modeled using heat resistors between the lumped capacitances. If necessary, the model can be refined by adding heat capacitances to better model reality.

Bond graphs can be used to construct heat flow models in two ways [67]. As bond graphs represent energy flow regardless of the type of energy, they can be used to model interactions between the thermal domain and other domains, for instance mechanical or electrical. The convention for a bond graph is that the product of the two variables

associated with a bond must equal power. For example, in the mechanical domain the variables are Force and Speed; in the electrical domain they are Voltage and Current. For the thermal domain, these variables should be Temperature and Entropy flow. This choice allows direct interaction with other domains. However, with this choice the capacitances and resistances of the thermal domain will be non-linear, making them complicated to model.

It is also possible to model heat flow using linear elements with Pseudo bond graphs. Here, the two bond variables are Temperature and Heat flow. As heat flow is a form of energy flow, this choice of variables means that the product of the two bond variables does not equal power, hence these models are called *pseudo* bond graphs. Direct connections between domains are not allowed in pseudo bond graphs. However, as the transformation of energy between domains is not directly relevant for a printer, pseudo bond graphs are well suited to make lumped thermal models of a printer.

7.3 Modeling heat flow with pseudo bond graphs

A heat flow model using pseudo bond graphs usually consists of the following elements:

- Heat sources. These inject heat into the model, and represent e.g. electrical heating elements.
- Heat capacitances. These stores heat, and represents physical mass.
- Heat resistances. These forms the connection between two heat capacitances. The amount of heat that flows depends on the temperature difference between the two masses and the resistance, which is dependent on e.g. material and construction.

An storage element analogous to the electrical inductor is not needed for heat models, as there is no physical effect that would require it for accurate modeling.

Using these elements, and the usual 0 and 1 junctions, a model can be constructed (see [20]). In Figure 7.2, a bondgraph model of a simple heater is shown. The bondgraph model is equivalent to the iconic model shown left in Figure 7.2; the simulation tool 20SIM supports both types of models.

7.4 A case study

7.4.1 Modeling goal

The goal of the model described in this case study is to be able to predict the time in which the printer is ready for printing, and what happens during the first minutes of printing when all temperatures settle to the new situation. A printer is ready for

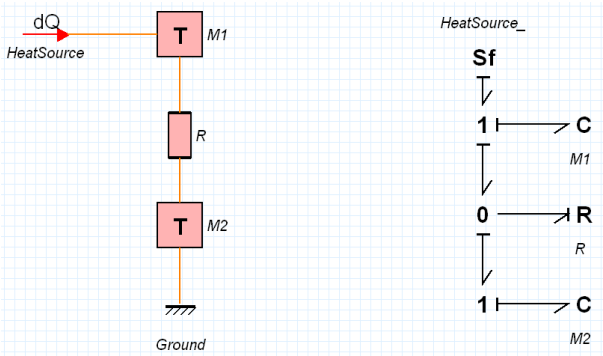


Figure 7.2: A bondgraph model

printing when it has been warmed to such an extent that the fuse temperature stays within acceptable bounds if printing starts at that time. Thus the following phenomena should be modeled:

- The heating of the parts that determine the fuse temperature. These are mainly the pre-heater (VVW) and the rubber TTF belt.
- The heating of paper sheets in the pre-heater.
- The exchange of heat between the TTF and sheets during fusing.
- The main elements that extract heat out of the TTF (various rollers and other losses to the environment).

7.4.2 Modeling strategy

The following simplifications have been made:

- The printer is modeled in one dimension: two or three dimensional distributions have largely been ignored. However, it is taken into account that the TTF does not have a uniform temperature: just before the heater, the TTF surface temperature is the lowest, just behind the heater it is at its highest. From the base temperature of the TTF and the amount of heat put into it, the temperatures at other locations are calculated.
- When interacting with the TTF, sheets are not modeled individually but as a continuous heat flow.

Using these simplifications, it is expected that the fuse temperature can be predicted with sufficient accuracy.

7.4.3 The model

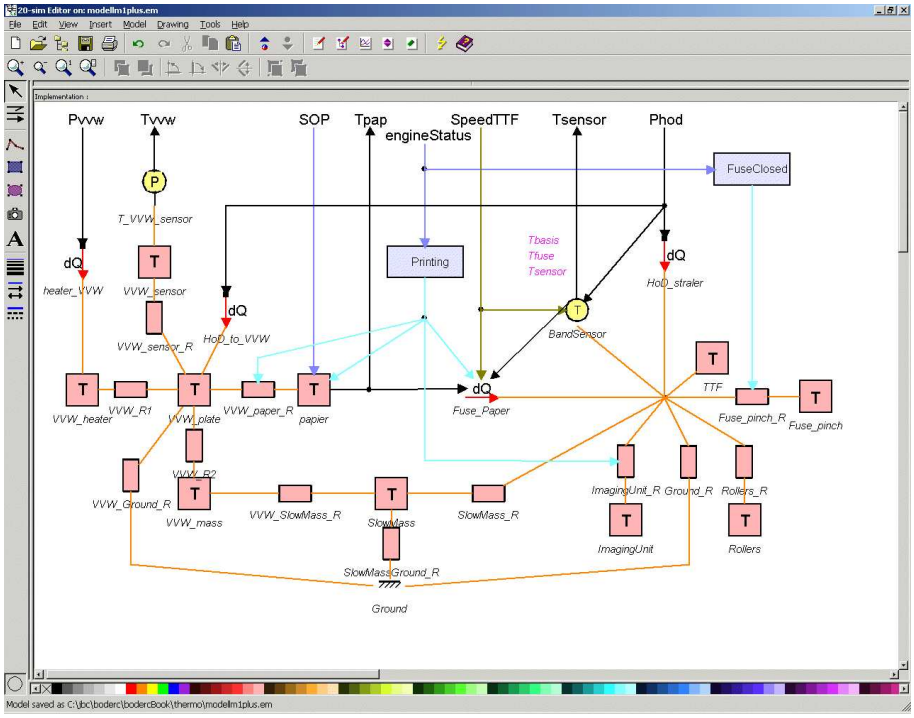


Figure 7.3: Printer engine submodel

Based on these assumptions, a model can be made of the printer heat flow. The tool 20SIM was used to construct the model. The heart of the model is shown in Figure 7.3. The model is based on the temperatures of the TTF-core and the pre-heater (‘VVW_plate’). From these locations, connections are made to:

- The heaters that insert heat to the system: HoD_straler and heater_VVW.
- The sensors T_VVW_sensor and T_sensor.
- The various rollers that are in contact with the TTF.
- The paper that extracts heat from the pre-heater and exchanges heat with the TTF.
- The environment to which heat is leaked.

The rotational speed of the printer is an input to the model. This is because the heat flow through a rolling contact is dependent on the roll speed. Through experiments, the following relationship was found:

$$R = \frac{c}{\sqrt{v}}$$

where R is the heat resistance; v is the roll speed, and c is a constant depending on material characteristics and construction details, that is determined experimentally. In 20SIM, this formula is easily inserted in the heat resistances.

Many of the contacts are dependent on the mode of the printer. For example, to prevent unnecessary heat loss, rollers like the imaging unit are only connected to the TTF during printing, not during warm-up. This effect is modeled by giving the heat resistance between TTF and imaging unit a second input, besides the rotation speed, that determines whether there is contact or not.

The contact between TTF and paper was hard to model. The paper is first heated in the pre-heater, before the contact with the TTF is established. To model the heating, a resettable integrator is used that is reset at the start of each page, using the Start Of Page (SOP) signal. If a normal capacitance / resistor combination is used, unwanted interactions between VVW and TTF would result. Thus the paper / TTF interaction is modeled by a power source that is driven by the final paper temperature (Tpap).

The model shown in Figure 7.3 is not self-contained. There is an interface to a higher level that determines the amount of power inserted into the printer, the input variables printer status and printer speed. Due to this structure, the model can be used for simulation but also for validation. To validate the model, earlier measurements can be inserted into the model using the variables, as shown in Figure 7.4. During simulation, a controller can be implemented that uses the temperatures generated by the model to control the virtual printer.

7.4.4 Identification and validation

The model includes various parameters that need to be specified to reflect the design choices made. Some of these parameters can be calculated analytically, but most of them need to be identified from measurements.

Obtaining measurements for identification is not a trivial exercise. To be useful, measurements need to have the following characteristics:

- The thermal conditions of printer and environment need to be known accurately.
- The state of the printer needs to be known accurately. For this a log needs to be maintained of the changes to the prototype.
- The relation between measurements and simulation results needs to be as clear and simple as possible.

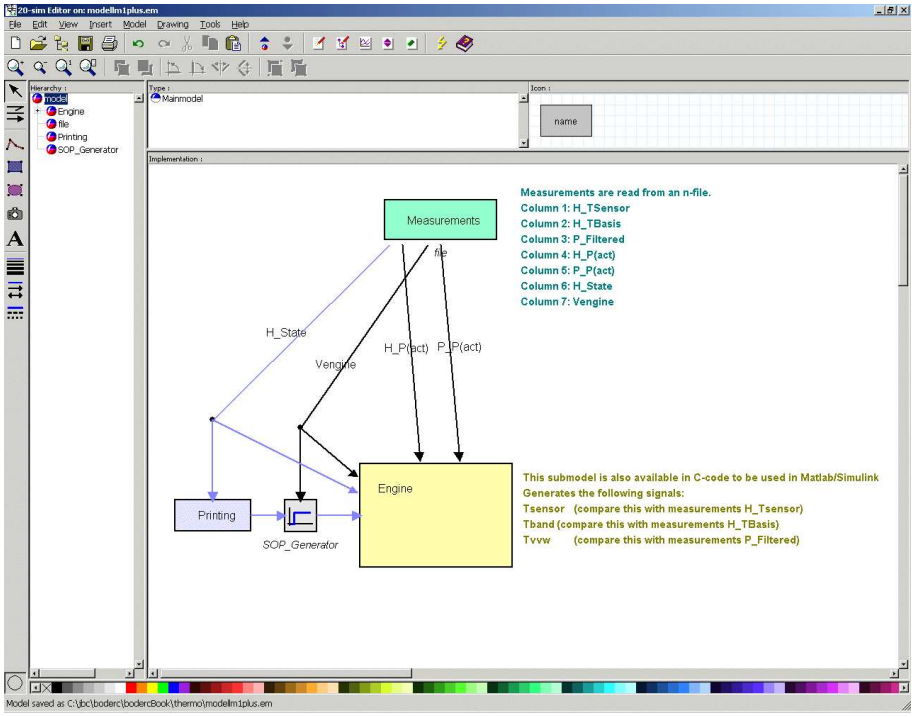


Figure 7.4: Top-level model

- Only one parameter should be changed during the experiment, preferably using a simple profile, e.g. a step or sinus profile. This enables the modeller to see if the model structure is sufficiently accurate.

- The behavior and configuration of the control software inside the prototype must be known exactly.

- The state of mechanical switches in the printer must be known at all times. These are determined by the software controling the prototype, which will frequently change.

As thermal experiments have a long duration, care must be taken that the printer is not disturbed during the experiment. Also, the measurements must be continuous during the experiment. Measuring should not stop when the printer is shut down.

7.4.5 Managing multiple experiments

When performing measurements on a prototype, it is important to be able to compare different measurements with each other. A model is always a simplification of reality. Thus, there will always be differences between model and reality. By comparing modeled and measured temperatures for many experiments, it can be evaluated if the 'typical' case is correctly modeled.

An advantage is taken from the separation of the control model from the physical model. The state of switchable contacts and the power injected into the printer is measured, and are injected into the physical model. Then, the outputs of the model are compared with the measured outputs, and parameters are tuned to get a proper response.

A scripting language is needed to do this effectively. It must be possible to quickly modify C and R values, evaluate the models quickly in a batch process, and present the results in a way that gives overview. At the time of writing, 20SIM did not offer a scripting language. Therefore, C-code was generated automatically from the detailed model, and this C-model was evaluated using the Matlab scripting language in Simulink. Figure 7.5 shows how the 20SIM model was encapsulated in Simulink. Figure 7.6 shows the top-level Simulink model.

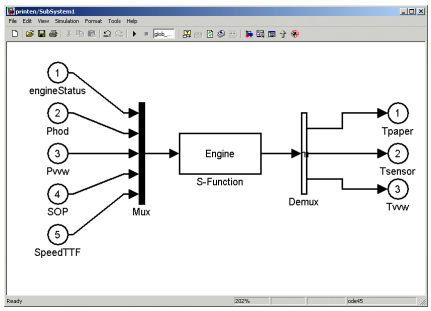


Figure 7.5: 20SIM model in Simulink

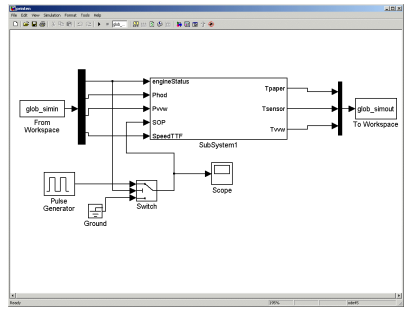


Figure 7.6: Top-level Simulink model

This set-up allows the evaluation of the model with measurements taken in different conditions, to see how robust the model is. If the evaluation shows that the model is not accurate in different conditions than those used for identification, the model is an oversimplification. Missing elements can then be added to the model. Also, this set-up gives the possibility to implement automatic tuning of model parameters. However, in the current implementation this is not present yet. This is a topic for future research.

7.5 Conclusions

As shown in this chapter, heat flows can be modeled using 20SIM. However, identification of the model parameters from experiments takes considerable effort.

At the time of writing this chapter, the model was still under development. Thus no definite results can be given. However, the expectation is that using this model, the following can be achieved:

- Optimization of the warm-up procedure.
- Testing and tuning the control algorithms for temperature.
- Optimization of critical parts of the heat flow design.

It is expected that it will be straightforward to re-use this model for other printers of comparable design. Also, the modeling strategy described in this chapter can be used to construct heat models for any printer. It is also expected that by using the model described in this chapter, the design time needed to optimize the heating of a new printer can be substantially reduced.

Chapter 8

Modeling of performance

Authors: P.F.A. van den Bosch, O. Florescu, M.H.G. Verhoef and G.J. Muller

8.1 Introduction

The performance of the control system is an important aspect of a machine. It would be a waste if a high-tech machine has been build such that it can physically achieve a high throughput, for example printed sheets of paper, but is limited because the software controlling it cannot keep up. Unfortunately, with current techniques it is hard to 'predict' beforehand what the performance of the software will be when it finally runs in the real system on the real processor(s). There are two (extreme) ways to deal with it:

1. Over-dimension the hardware platform to make sure the software will run.
2. Implement the software, then run and evaluate its performance on the target hardware platform. Then use this information in the next design cycle.

The disadvantages of both approaches are clear. In the first situation the cost price of the entire system will surely be higher than necessary. In the second case, the design time is increased dramatically because more design cycles are needed. Therefore, it is important to strive to a development method that leads to fast design cycles for software performance, while having an accurate enough prediction.

8.2 Problem formulation

As explained, the goal is to find or develop methods, techniques and tools that make it possible to predict the performance of software accurately based on only a small model

that does not need a lot work to come up with. Obviously, there is a tension between the accuracy of the performance prediction and the amount of work needed to make the model. In general, it is even likely (but not proven) that it will require more work to make an *exact* prediction of the performance than it would be to create the whole system, run it and see how it performs.

During the process of performance modeling, and also during other Boderc activities, we realized that the goal of creating a model is not only to do an analysis and to make a prediction. Probably more important is the understanding that is obtained by creating the model. This understanding leads to the ability to make better design choices and to be able to understand the influences faster, thus decreasing the design cycle time.

Summarizing, the aim of this work is

> *A model of the performance characteristics of a control system that increases the understanding of the relations between hardware and software parameters, such that in early design stages enough confidence is gained to be able to iterate through the design choices with a short cycle-time.*

8.3 Modeling approach

In this chapter an approach is presented to make a model according to the aim mentioned before. Although there are techniques, like the ones presented in [117] and [40] that enable analysis and prediction of the performance of a system before its actual realization, they are not largely used in industry because of their conservatism or problems to scale with the dimension of the system. Each method makes a trade-off between the time spend to make such a model and the accuracy of the results. Many things influence the performance of a system. In Figure 8.1 an overview is provided of typical factors that determine the performance. Four layers are considered:

- The lowest level is the *hardware platform* that influences the performance through processor speed, bandwidth and access latency. The efficiency at which it can make use of the memory bandwidth is increased by a memory cache. However, this makes the performance less predictable and more dependent on what exactly runs on the processor.

- The next layer is typically the *operating system*, including a scheduler, which takes care of resource sharing by handling task switches and interrupts, and can provide advanced inter-process communication.

- Then there might be another layer, the *middle ware* or services that typically provides *services* and abstractions.

- The top-layer is the application itself. This application might be modeled entirely with the help of the middle ware layer, but usually also contains direct RTOS calls and might directly access the hardware.

The performance of the entire system depends on how the higher levels use the lower levels. On the vertical bar in Figure 8.1 the *tools* are mentioned, like compilers and linkers but also code generators of the middle ware that can have a huge influence on the performance.

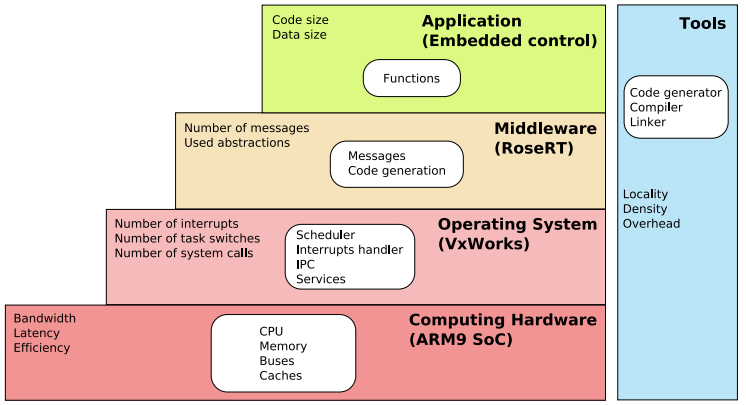


Figure 8.1: Important layers when considering software performance.

The modeling approach is to consider these layers and to characterize the important aspects of all these layers with quantifiable parameters. Ideally, the model will be a formula in which the performance (execution time) of the application is expressed as a function of the middle ware, RTOS and hardware parameters. The middle ware again can be expressed as a function of the RTOS and hardware and the RTOS as a function of the hardware alone. Unfortunately, some characteristics on the lower levels are dependent on the higher levels. For example, the efficiency of a processor is boosted by the use of caches, but the higher levels and tools determine what the influence of the cache will be. Despite it is hard or impossible to estimate these influences accurately, it will be shown that it is possible to create useful insights in the performance.

8.3.1 The case under study

In the next paragraphs, the embedded control software of a printer / copier will be taken as a study object; it wil be used for measurements and modeling. The embedded control consists of roughly two parts: a hard real-time part and a soft real-time part. The hard real-time part is the lower level that takes care of things like motor controllers, heater controller, and paper transport; it directly interacts with the environment. The higher layer (soft real-time) is in charge of planning: it receives requests to print or scan one or multiple pages and then makes a detailed planning for these sheets. The planning considers the availability of all functions, like paper path, finisher and printing process. Once this planning or allocation is ready, it is communicated to the lower level control, which will execute it and report back on success or error conditions.

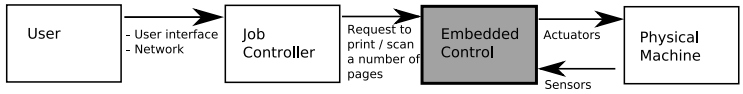


Figure 8.2: Positioning the embedded control in the printer.

In our casestudy, the control software runs on a microprocessor (ARM9) on which the VxWorks operating systems is also running. The aforementioned hard real-time tasks are all executed in a periodical task that is called every 2 ms. This task has a high priority to make sure its behavior is very predictable. The other tasks (like allocation, error handling et cetera) run as VxWorks threads with lower priority. Most of the control software is generated from RoseRT and uses an extra abstraction layer, the RoseRT runtime system. This runtime system (RTS) includes a mechanism to handle messages between capsules (objects) and handles the execution of state machines that are part of the capsules. The RTS and the application can be spread over multiple threads (each capsule has its own thread) or combined in one.

So, when the system is running, the hard real-time task will interrupt the other tasks every 2 ms and run until completion (of course much less time than 2 ms). The other tasks will only run in the processor time that is left, and typically take longer to finish.

8.4 Characterization of the layers

As proposed, the model will be a function that relates the performance of the application to the other layers. For each layer it is possible to measure or calculate a few characteristics. These characteristics can be used to evaluate the performance of the control software as a whole.

8.4.1 Characterization of the hardware platform

Figure 8.3 shows the architecture of the chosen system-on-chip (SoC) with ARM9 core. The CPU core runs at a maximum speed of 200 MIPS (Million Instructions Per Second), but because the latency and bandwidth of the memory is much slower this speed will only be reached when all instructions and data are in cache. The system has

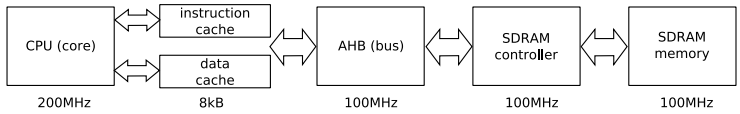


Figure 8.3: Simplified structure of the SoC showing parts relevant for code execution.

a two-level memory hierarchy, with a level-1 instruction and data cache and external SDRAM. The cache has 8 words per cache line and 4 sets of 64 cache lines each,

resulting in 8kB for instruction and data cache separately. The SDRAM memory and controller have a maximum bandwidth of 100 MHz. Figure 8.4 distinguishes external and internal latencies. Internal latencies are between the CPU core and the SDRAM controller, external latencies are between SDRAM controller and the external memory. In the case of a cache miss, whole cache lines are fetched at once, which leads to an additional transfer time from memory of 8 memory clock cycles. The CPU-core includes a five stage execution pipeline, the third stage is the execution stage.

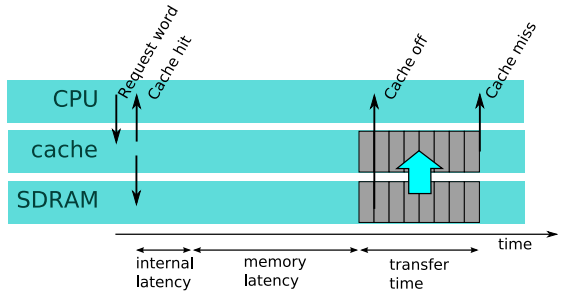


Figure 8.4: The time for fetching an instruction varies depending on cache setting and availability in cache.

Equation (8.1) is a simple formula for the time it takes to execute a piece of code.

$$T_{exec} = N_i \cdot T_{cpu} \cdot CPI \tag{8.1}$$

where

N_i	Number of instructions in piece of code
T_{cpu}	One CPU cycle $\frac{1}{f_{clk}}$
CPI	Average cycles per instruction.

The formula can be further refined by specifying the average *CPI* more accurately. When instructions and data are available in the cache, the *CPI* of that instruction will be equal to the one specified in the data sheet of the CPU. Depending on the instruction, it will take 1 to 3 CPU cycles. A branch, for example, typically takes 3 clock ticks because the contents of the pipeline becomes invalid. When the cache does not contain either the instruction or the data (or both), the CPU will be stalled until it is available. Fetching from memory is slower, because the memory bus is slower, with a factor N_{div}, than the CPU clock. Accessing the memory results in an additional latency; this latency includes amongst others the so-called CAS-latency and is in total N_{lat} memory cycles. Formula (8.1) can be refined by splitting the instructions, N_i, in instructions that are in cache, N_{fast}, and instructions that are not in cache, N_{slow}.

$$T_{exec} = N_{fast} \cdot T_{cpu} \cdot CPI + N_{slow} \cdot T_{mem} \cdot (N_{lat} + N_{penalty}) \tag{8.2}$$

where

T_{mem}	One Memory cycle $T_{mem} = T_{cpu} \cdot N_{div}$
N_{div}	Factor between memory and CPU speed

cache setting	measured time [CPU cycles]
normal	815 to 3.9k
flushed	3.9k
off	18k

Table 8.1: Measured execution time for 800 NOPs with different cache settings.

The penalty time, $N_{penalty}$, will be explained later on. In order to measure those (combinations of) latencies, 800 individual instructions (eg NOPs) are executed multiple times. This program can be run with different settings of the cache. When the cache is on, eventually all instructions will hit in the cache. This results in a hit rate of 100 %. When the cache is flushed before the execution of the program, all the instructions have to be fetched again (8 at a time, so 100 fetches) from memory. Effectively, this results in a hit rate of 87.5 %. When the cache is disabled, it needs to fetch all instructions (800 times) from the memory separately. This corresponds with a hit rate of 0 %. The resulting execution times for the different situations are measured and listed in Table 8.1.

Figure 8.5 shows how the instructions are fetched and executed for different settings of the cache. It is shown that executing instructions is done parallel to transferring them from memory to cache. When all fetches hit in the cache (1 in Figure 8.5), an instruction is executed every CPU clock, there are no latencies. In the case that the fetch initially misses, the instruction is fetched together with 7 other instructions (2). As soon as the first one is in the cache, it can be executed (3), the latency is 23 CPU cycles. The next sequential instruction can only be executed when it is transferred from memory that is why it is 1 memory clock cycle (2 CPU clocks) later. When the next instruction results in a cache miss, it is still necessary to complete the transfer of all 8 words before fetching of the next words takes place (4), in this case the effective latency adds up to 38 CPU cycles. In the case that the cache is disabled, a word is always fetched from memory before it can be executed (5), the delay is always 23 CPU cycles.

From the measurements and equation (8.2), it follows that the latency, N_{lat}, is 23 CPU cycles (or 11 memory cycles). $N_{penalty}$ is used to deal with the different effective latency in the case that not everything is in cache. If the hit rate is $\frac{1}{8}$, only one instruction is executed while 8 have been fetched, the penalty in that case is $8 \cdot T_{mem} - 1 \cdot T_{cpu}$. However, if the hit rate is $\frac{7}{8}$, the penalty is $8 \cdot T_{mem} - 7 \cdot T_{cpu}$, because those 7 CPU cycles were effectively used to execute 7 instructions in parallel with transfers from memory to cache. In general: $N_{penalty} = 8 \cdot N_{div} - 8 \cdot HR$, with *HR* the hit rate.

Note that it depends largely on the type of instructions what the average *CPI* is. For example, instructions are only executed efficiently if the code is sequential without branches. A branch instruction flushes the pipeline and has to wait for the cache line to be filled entirely. For now, the effect of the 5-stage pipeline is neglected: an instruction is assumed to be executed when it is available.

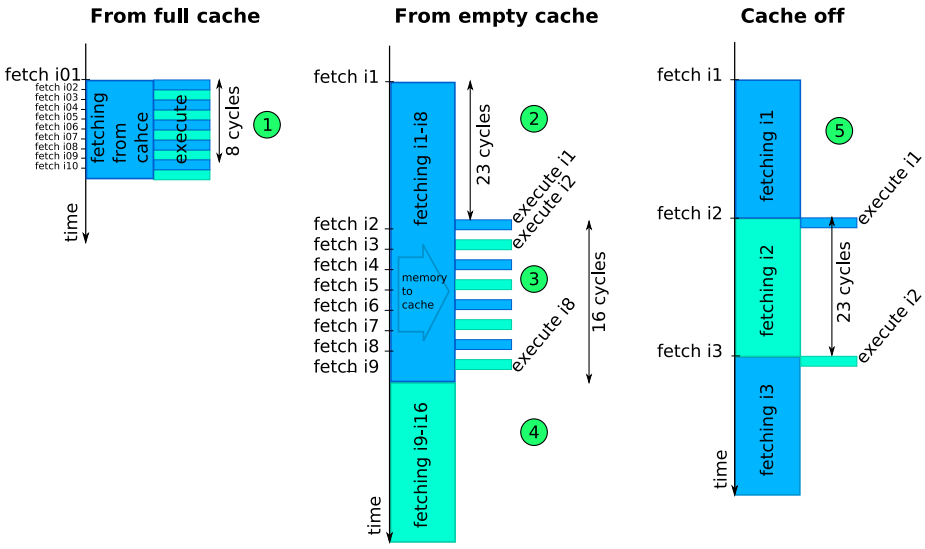


Figure 8.5: Timing for fetching and executing instructions with caches disabled and enabled.

Measurement method

For all the timing measurements, an on-chip timer has been used. This timer has a resolution of 270 ns. From a few tests of reading the timer register, it has been concluded that the accuracy of the timing method is 200 ns (40x 200MHz-cycles).

Assumptions

In order to simplify the formula, many assumptions were made. These assumptions are important because if they do not hold or cannot be neglected, the formula does not hold and needs adaption. The most important assumptions are:

- Extra latencies caused by the SDRAM are deemed irrelevant. For example switching banks in the memory chips results in higher latencies, but data and code have their own memory banks, and most code is assumed to be very local, reducing jumps over bank boundaries and over SDRAM rows that are 256 words long.
- The pipeline of the CPU does not stall, this means no branches (sequential code) and no instructions that have to wait for each others data. When this is not the case, the average *CPI* will increase, but also the penalty will be different.

8.4.2 Characterization of the RTOS

The RTOS, VxWorks, provides a scheduler that activates and deactivates tasks based on their priority. The scheduler is invoked periodically by a timer and sometimes by tasks through system calls like suspend and semTake. Every time the scheduler is invoked, it has to determine which task to run next and this involves context switching: store the state of the previous task and load the state of the new task. Typically, a profiler like WindView does not show this overhead: it only shows when a task 'ends' and apparently the next task immediately starts. With two tasks, like in Figure 8.7, it is possible to measure the task switching time. Figure 8.6 shows this graphically: two tasks exist that both run periodically, the timer is read before the suspension of task 1 and after the suspension of task 2, which runs at a lower priority. As soon as the main task suspends, task 1 will resume, the cache flush is performed, the timer is read and task 1 is suspended, after which the previously suspended task 2 resumes.

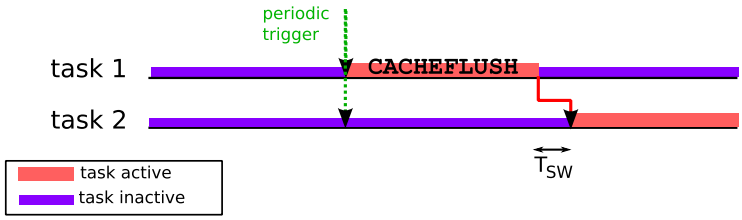


Figure 8.6: To measure the task switching time, we use two tasks that execute sequentially with a cache flush before the switch.

The results of the measurements are shown in Table 8.2. Typically a task-switch will take between 1.6 (best-case) to 20 (worst-case) μs, when caches are enabled and depending on whether the code between the task-switches messes up the cache a lot. According to Table 8.2, a task switch with cache disabled takes 10k CPU cycles. 10k divided by 23 cycles per instruction (see Figure 8.5) is 430 fetches, both instructions

```
/* High priority */
void task1( void)
{
  while( 1)
  {
    CACHEFLUSH;
    READ_TIMER( startTime);
    taskSuspend();
  }
}
```

```
/* Low priority */
void task2( void)
{
  while( 1)
  {
    taskSuspend();
    READ_TIMER( endTime);
  }
}
```

Figure 8.7: Example of code used for measuring the task switching time.

cache settings	T_{sw} [CPU cycles]	[μs]
normal	320 to 1.6k	1.6 to 8
flushed	2.8k to 4.0k	14 to 20
off	9.4k to 10k	47 to 51

Table 8.2: Measured task switch time for different cache settings.

and data. Best case 320 CPU cycles are needed, which means that it mostly runs from cache!

Caching effects by context switching

When a task is interrupted by another task, the current content of the cache is typically worthless: different code will be executed. First the scheduler of the RTOS and then the next scheduled task will be executed by the processor; the cache needs to be 'refilled' with relevant contents. Knowing the size of the cache it is possible to estimate the worst-case effect. At most 256 cache lines must be refilled, which gives an overhead of 39 CPU cycles per line: $256 \cdot 39 \cdot 5ns = 50$ μs. Therefore, it can be argued that penalty caused by the pre-emption of a task is 50 μs.

8.4.3 Characterization of the middle ware: RoseRT

Approximately the same measurement as done for VxWorks with the context switch can be done for RoseRT. Instead of tasks, capsules are considered that send a message (an integer) to each other, see Figure 8.8. Before sending the message with `messageOut.signal1(0).send()` and after receiving it with `MessageIn`, a time stamp is taken.

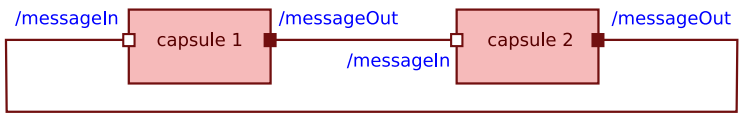


Figure 8.8: Two capsules that send messages to each other.

The scheduling of capsules and messages is done by the RoseRT runtime system, which is linked together with the application code. It can be chosen to make a physical RTOS thread for each capsule, or to map them both on the same physical thread. Depending on this choice, the overhead is different, as shown in Table 8.3.

Physical threads	Cache	Latency
one	normal	[6, 37] μs
	flushed	[33,43] μs
separate	normal	[28, 67] μs
	flushed	[82, 98] μs

Table 8.3: 'Overhead' of sending a message between capsules in different configurations.

8.4.4 Characterization of the application

Formula (8.2) can be refined more by taking the hit rates of the caches into account, as in formula (8.3). $N_{penalty}$ has been replaced by its value depending on the hit rate.

$$N_i \cdot ((19 \cdot T_{mem} - 8 \cdot T_{cpu} \cdot (1 - MR_i)) \cdot (1 - MR_i) + CPI \cdot T_{cpu} \cdot (1 - MR_i)) + N_d \cdot ((19 \cdot T_{mem} - 8 \cdot T_{cpu} \cdot (1 - MR_d)) \cdot (MR_d)) \tag{8.3}$$

where

N_i	number of instruction fetches
N_d	number of data fetches
T_{mem}	memory clock cycle time
T_{cpu}	cpu clock cycle time
MR_i	cache miss rate for instructions
MR_d	cache miss rate for data

Therefore, a piece of code (program) can be characterized by values for MR_i, MR_d, N_i, N_d, and *CPI*. The values for T_{mem} and T_{cpu} are hardware characteristics. For an existing application, the cache miss rate can be measured by executing the code and measure the execution time with caches enabled and again with caches disabled for both data and instruction cache separately. That will result in 3 measurements, obtaining 3 equations for the parameters. Unfortunately, there are 5 independent variables. However, it is possible to determine the set of possible solutions.

The measurements for three cache settings were performed for a part of the soft real-time control code. After analysis, it turns out that there are 3 - 5 more instruction fetches than data fetches. Furthermore, the miss rate for instructions is between 0 and 5 % and for data between 0 and 18 %. Figure 8.9 shows the relation for different values of the *CPI*. The values near to 0 % can be confidently neglected, so *probably* the values will be around 3 % miss rate ($N_i = 7.7M$) for instruction fetches and a data cache miss rate of 10 % ($N_d = 1.8M$).

Usage of RTOS and middle ware

The overhead of the RTOS is mainly due to task switches; during a task switch, the scheduling function is executed. There are at least two task switches every 2 ms be-

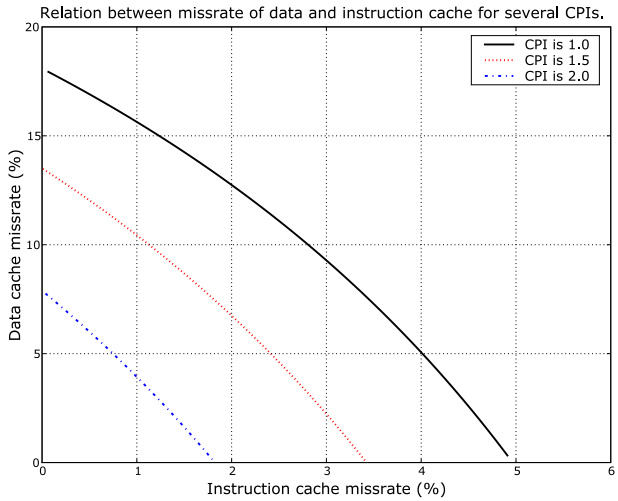


Figure 8.9: Miss rates of data and instruction cache as function of each other, actual value must be on this line, all based on measurements.

cause of the hard real-time task. Furthermore there are several other tasks, typically leading to 1500 task switches per second. This number hardly depends on the printing speed. The reason is that after the periodic task always another task is called. One task switch takes worst case 20 μs, the overhead by task switches is therefore at most $1500 \cdot 20 = 30$ ms per second, or 3 %.

The overhead of the middle ware is characterized in terms of message overhead. The amount of messages during a print job was measured (typically, this can be done on the target platform if available, but just as well on a simulation on the host). The number of RoseRT-messages per page is 210. Of these messages, 120 are internal in a thread and 90 are between threads, causing extra overhead. With the help of Table 8.3, the maximum overhead caused by the messages is calculated to be $120 \cdot 43 + 90 \cdot 98 =$ 14ms per page. Suppose the printer has a speed of 60 pages per minute, then the overhead is at most 1.4 %.

Additional influence of cache

As explained earlier, due to task switches, the cache is spoiled which makes the interrupted task less efficient. In this system, an interrupt occurs every 2 ms, flushing the cache. When a cache is spoiled, it will take at most 50 μs to refill all cache lines and make the interrupted task run at full speed again. In this case, the harm done by this flushing is therefore *at most* $500 \cdot 50 = 25$ ms per second, thus 2.5 % CPU time. This is the effect of periodic interruption on the soft real-time tasks.

Speed cpu,mem	Estimated time (CPI=1.0)	(CPI=1.5)	Measured time
200,100	107 ms	107 ms	108 ms
100,100	137 ms	161 ms	159 ms
200,50	184 ms	160 ms	178 ms
180,60	162 ms	148 ms	-
160,80	134 ms	134 ms	-

Table 8.4: Predicted and measured execution time at different clock speed configurations.

8.4.5 Validation

In the previous section, formula (8.3) was shown that claims to predict the execution time of an application based on a few measurements on the bare level. With these characterizations, the effect of changing hardware parameters can be estimated. It has been shown already that the effects of task switches and messages can be neglected, although the effect of the parameter changes can also be calculated for them. The effects of four additional hardware platforms are considered (see Table 8.4): the same SoC but with other clock rates for CPU and memory. For two configurations, the measurements are also done for validation. For the configurations of 180 MHz CPU and 60 MHz memory bus, and 160 MHz CPU and 80 MHz memory bus, no validation is done, only a prediction. The latencies of the memories are kept the same number of clock ticks for all configurations. Table 8.4 shows the measurement results and the corresponding predictions from equation (8.3).

It is clear that the correctness of the answer depends highly on the *CPI*. During the previous analysis, a method to correctly estimate the *CPI* has not been considered, but it turns out to be very relevant for the prediction of the execution time.

8.5 Conclusions

In the problem formulation we stated that we wanted to come up with a simple model to estimate the performance of the embedded control software. In the following sections some formulas and measurements have been given. As was already said in the problem statement, one of the most important aspects of making a model or a formula, is the insight gained from the formulation. Making a model forces the engineer to be explicit and to quantify and measure relevant aspects, like for example the number of task switches. This is exactly what can be concluded from the case study: insight was gained, but a simple formula that can accurately predict performance on a chosen platform is not yet available. Additionally, the following is concluded:

- A method has been proposed to create a model to estimate the performance of an embedded software application. It is proposed to do simple measurements at

each layer. In the particular case, the overhead that can be expected by RTOS and middle ware is limited, it is only a few percent. When going to a higher printing speed, only the middle ware introduces *additional* overhead, but it will only become significant at very high printing speeds.

- The method to link application performance to hardware characteristics does provide a lot of insight in the processor workings. It also gives insight in estimates of characteristics of the application, like cache miss rates and number of instructions. However, the validation shows that especially the *CPI* is a crucial parameter that has not been addressed thoroughly enough yet.

- In this particular case it has been shown that the overhead introduced by using messages of RoseRT is not very much, approximately 2 % of the total. The same argument holds for the time 'lost' in context switches. However, in new cases these aspects must definitely be measured and calculated again, it is the only way to be sure.

Furthermore, we like to make the following remarks and recommendations:

- When moving to another platform than the current ARM9, the application itself is not going to change much. However, the execution times will differ. Take for example a Pentium processor. The execution speed of the core is much higher than of the ARM, a factor 10, 2 GHz instead of 200 MHz. The memory bus is typically faster with respect to possible sustained throughput, typically 400 MHz. However, the latency of the memory is not less, it might be even more because of the complexity of a Pentium board, there is a bridge between processor and memory, which will increase the latency. On the other hand, a Pentium has a large L2 (even L3 cache) in which very large parts of the code can reside. The chance that these caches have a miss are very small. Anyway, what needs to be done are the micro-measurements, to get a feeling for the latencies and speeds of the processor board. The effect of the different caches has to be measured and taken into account, this means that it is necessary to estimate the cache misses for all three caching levels.

- There are numerous 'details' that influence the execution time of a piece of code. Some of them are parameters of the formulas and can be varied to study the effects. But other things like compiler flags are not in the formulas, but they *do* influence the execution time. It is important to carefully keep track of all of them, to make them *explicit*. It would be a good idea to generate a list of relevant parameters to consider. An engineer can then take this list and pick the relevant items for his particular problem.

- Even if information about latencies and bandwidths is available in data sheets or given by another designer, it is worthwhile to do a few measurements. This will give a better 'feeling' and forces to validate the implicit model.

Chapter 9

Virtual printer modeling

Author: K.J. Klein Koerkamp

9.1 Introduction

From a customer perspective, print quality is an important aspect of a printer. This was already discussed in Chapter 3 where print quality was identified as one of the key drivers of printers. In order to improve the print quality of Océ's printers, it is important to know what the effect of the properties of a printer is on the resulting print quality. Such knowledge would support taking decisions in an early phase of development of new printers. Virtual printer models have been developed to gain insight in the complex relations between technology variables such as resolution, dot size, ink colors, et cetera, and print quality. Another goal of the use of virtual printer models is to predict what the print quality of new printer concepts will be. Often it costs a considerable amount of effort to design and manufacture a test-setup to test new ideas for a part of a printer. With the models the working and the effect on the print quality of this part can easily be evaluated. The virtual printer models can be viewed as 'plug-ins' (step 3b) of the high level method discussed in Chapter 2.

The framework for linking technology variables of a printer to the final print quality is given in Figure 9.1

The elements of the image quality circle are:

- *Technology variables* are the properties of a printer that are chosen during the design of a printer, such as the type of image processing, resolution, speed, dot sizes, ink colors, et cetera.
- *Physical image parameters* are the properties of a print that can be measured, such as optical density, dot positions, gloss, et cetera.

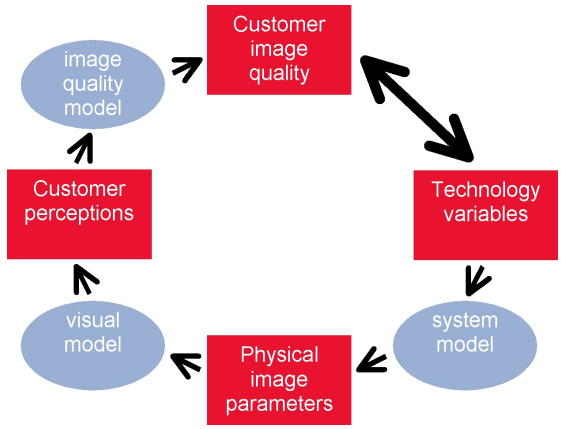


Figure 9.1: The image quality circle

- *Customer perceptions* are the perceived image quality aspects of images. Examples are color rendition, sharpness, detail visibility, text quality, area uniformity, et cetera.
- *Customer image quality* is the overall quality number a customer would assign to a print.

These elements are connected through a series of models:

- *System models* describe the behavior and the working of a printer.
- *Visual models* describe how prints are perceived by humans and how the perceived attributes depend on the physical image parameters.
- *Image quality models* describe how the different perceived attributes contribute to the overall image quality. This is often market and application dependent.

The virtual printer models that have been developed are system models that describe how the technology variables of a printer can be linked to physical image parameters of the final print. They simulate the behavior of a printer with certain settings, and the output is an image which represents the light reflection of a print. Currently at Océ there are no good visual models or image quality models. Linking physical image parameters to customer perceptions is done in the same way as is done with real prints: by a suitable measurement application or by visual assessment. Customer perceptions are linked to customer image quality by Océ's market experts or customer visits.

Océ has two color print technologies: direct imaging process (DIP) and inkjet. The printing process of these technologies is completely different. As a consequence, two different models have been developed to describe the process. Each model consists of several modules which describe the different process parts of a printer, and which

can be changed or improved independently. In this way experts of each process part can contribute to the model themselves by improving or adapting a module using their specific knowledge, without having to know much about other process parts. A project leader is needed to keep an overview over the different parts of the model. By developing the models in such a multi-disciplinary way, the virtual printer models have become a valuable developing tool, which can simulate printers with a high degree of accuracy.

The outline of this chapter is as follows. An overview of both virtual printer models will be given in the next two sections. After that some examples will be given of how the virtual printer models are used in industrial practise. This chapter will be finished with conclusions about the working of the virtual printer models and their benefit for printer development.

9.2 Virtual dip printer

Before describing the DIP printer model, first a description of the basic working of the printing process will be given.

9.2.1 DIP printing technology

From 2001, Océ successfully applies the unique DIP printing technology in its Color Production Systems (CPS). A global representation of the DIP process is shown in Figure 9.2.

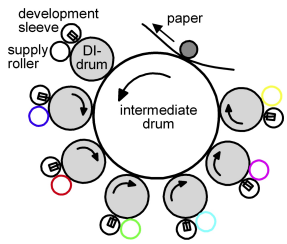


Figure 9.2: Schematic representation of the DIP-printing process

Seven color toner images are developed in seven direct imaging (DI) units, which each comprise of a supply roller, a development sleeve and a DI-drum. After development on the DI- drum surface, each single color toner image is transferred to the central intermediate drum. Finally, the total image is transferred and fused to paper in a single step applying pressure and heat. These steps will now be explained in more detail.

The first step is the development of the toner images in the DI-units. This process is illustrated in Figure 9.3.

The supply roller continuously develops toner particles on the surface of the DI-drum. The rotating DI-drum transports the toner particles towards the toner assembly in the development nip (the space between the development sleeve and the DI-drum). In

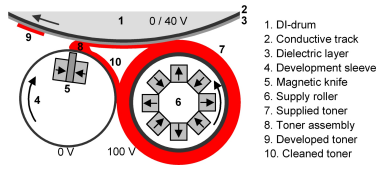


Figure 9.3: Schematic representation of the DI development process

the development nip two things can happen. The DI-drum consists of circumferential electrodes, tracks, which can be switched individually to a voltage of 0 V or 40 V. When a track has a voltage of 0 V, toner particles that enter the development nip on this track are magnetically pulled from the DI-drum to the development sleeve. They are transported back to the supply roller. In this way no toner is developed on this track. Whenever a track is switched to a voltage of 40 V, the electrical force on the toner particles on this track towards the DI-drum is larger than the magnetic force towards the development sleeve and the toner particles are not pulled from the DI-drum towards the development sleeve. In this situation, toner particles are developed on the DI-drum track. By switching the voltages of each track at the right moment, a toner image can be developed on the DI-drum. The axial and tangential resolutions of this image are determined by the track pitch and the timing of the print voltage per track. These resolutions are typically 600 dpi and 2400 dpi respectively.

Toner particles are always developed in clusters, because the size of a toner particle is much smaller than the width of a track. To develop a cluster, a track is switched to a voltage of 40 V for some period of time. The track will however not instantly be completely covered by toner particles. The distance needed to achieve full coverage is called edge sharpness. In Figure 9.4 it is shown how the toner-coverage cov(x) in the process direction x follows the print voltage V(x) for a cluster of toner particles with length a. Here the average toner-coverage profile is shown. The coverage profile of a single printed cluster would not look so smooth. Especially at the edges of a cluster there is much variation in the coverage. The length of the printed cluster is always larger than a. The extra length Δa is called line broadening. The shape and the variations in the shape of the individual clusters determine the final print quality to a large extend. A CCD-recording of developed cluster of toner particles on the surface of the DI-drum is also shown in 9.4.

After development, the seven color toner images are subsequently transferred from the surface of each DI-drum to the intermediate drum. Due to the adhesion forces on the toner particles, toner particles can not be transferred on top of other toner particles that were already transferred by other DI-units. This is illustrated in Figure 9.5: toner particles will only be transferred to ‘empty’ areas on the intermediate surface.

Finally, the full-color mono-layer toner image is transferred and fused (‘trans-fused’) from the intermediate drum onto paper. By applying heat and pressure, the toner particles are melted and pressed into the paper (see Figure 9.5).

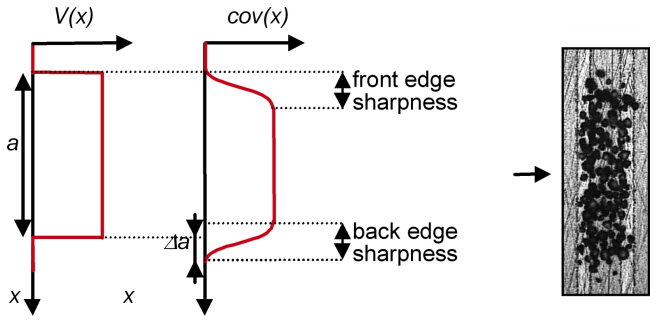


Figure 9.4: A print voltage signal on a track (left) is transformed into an average toner-coverage profile (middle). To the right, a CCD-recording of a developed cluster of toner particles on the surface of the DI-drum is shown.

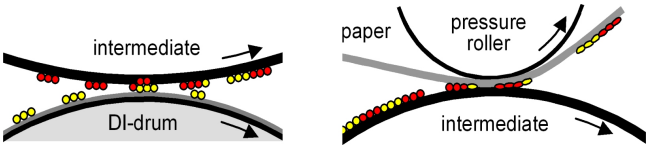


Figure 9.5: Left a schematic representation of the transfer principle, right a schematic representation of the transfer and fuse process

9.2.2 DIP printer model

The DIP virtual printer model is a system model that describes the working of the DIP printing process. The model consists of five main modules as illustrated in Figure 9.6: image processing, development, transfer, transfuse and the optical model. The five different modules of the model can be changed or replaced independently. The data flow between the modules consists of seven color bitmaps containing toner-coverage values per pixel. The final output of the model is a bitmap with color values for each pixel.

The digital image is processed by the image processing in exactly the same way as is done in real printers. The image processing generates an image for each DI-drum. This image contains the voltage level (0 or 40 V) for each track at each (discrete) moment of time.

In the development module the toner-coverage on the 7 DI-drums are simulated. A statistical model is used to predict coverage variations due to the complex interactions in the development nip. From measurements on clusters of toner particles on DI-drums, it is known that the front and back edge sharpness and the line broadening of the average coverage profile of a cluster are independent of the addressed cluster length a. The average values and the variations of these parameters are measured on a test-setup for a certain printer configuration, i.e. for different magnetic knife types, development

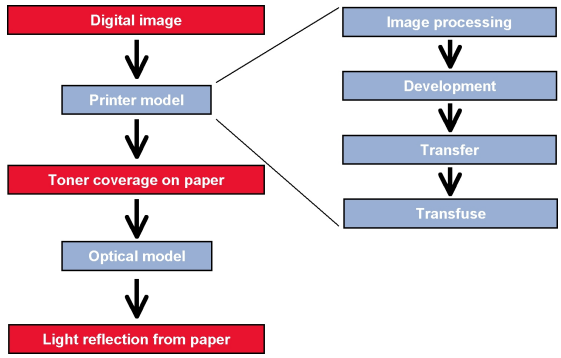


Figure 9.6: Flowchart of the DIP-printer model showing the different modules

sleeve speeds and toner types.

Now, it will be shown how this statistical model is implemented to calculate the toner-coverage profile of a single cluster. Usually a cluster has a length of several pixels (time moments). The coverage prediction process for pixel 2 within a cluster with a total length of 7 pixels is represented in Figure 9.7.

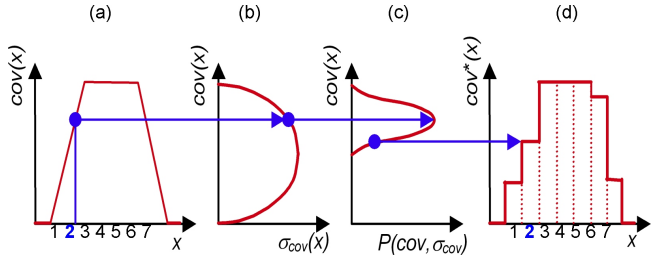


Figure 9.7: Statistical coverage selection process from average coverage profiles, to standard deviation, to normal distribution, to selected coverage profile representing toner-coverage per pixel on the surface of a DI-drum

First, the average coverage profile of the cluster is calculated, leading to an average coverage for each pixel within the cluster. The average coverage profile of a developed cluster is approximated by a trapezium profile, with the front and back edge sharpness and the line broadening as input parameters. Subsequently, the coverage variation is determined as depicted in Figure 9.7(b). Then, the corresponding normal distribution is calculated as illustrated in Figure 9.7(c). From this distribution a coverage $\text{cov}^*(2)$ is taken for pixel 2, which represents the fraction of the area of the pixel that is covered with toner particles. The end result after evaluation of all pixels within a cluster is a coverage profile for the evaluated cluster (depicted in Figure 9.7(d)). For each of the seven colors, a bitmap is filled with coverage values representing toner-coverage for all

pixels on the surface of each DI-drum. The development module has been validated by comparing simulations with measured cluster coverage profiles on the DI-drum surface using a high speed CCD camera. An example of such a validation result is shown in Figure 9.8. It is shown that the assumed statistical model with the front and back edge sharpness, the line broadening and the standard deviations of these values as input, is sufficient to predict the coverage profiles and the coverage variations of clusters in good agreement with the measured coverage profiles and coverage variations.

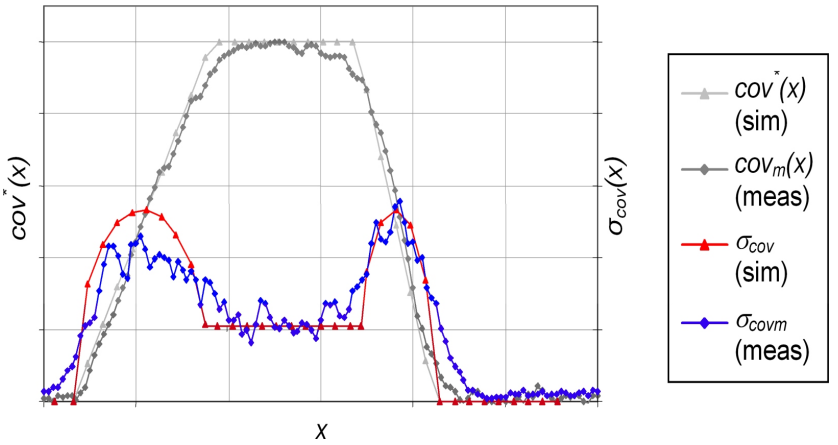


Figure 9.8: Validation of the simulation of toner development on the surface of the DI-drum

The development module also incorporates a similar statistical model to simulate the coverage profiles in the axial direction arising from the transitions from track to track.

In the transfer module of the model, the seven bitmaps with coverage data are transferred to the intermediate drum in the order as applied in the printer. Because the black toner image is the first to be transferred, the probability that toner particles are transferred from a pixel on the DI-drum to a corresponding destination pixel on the intermediate surface is 100 percent. However, the transfer probability per pixel of the following color toner images depends on the toner-coverage that is already present at the destination pixel, i.e. the sum of previously transferred toner-coverage. Other input parameters regarding this module are color-to-color registration errors and variations in transfer efficiency.

In the transfuse module, toner particles are considered to be homogeneous spheres which are pressed into a disc-shape during transfuse. This results in a coverage increase and a layer thickness decrease. In general, the coverage increases with about 33 percent, depending on the toner and paper type and the transfuse pressure and temperature. This module's output are seven bitmaps containing coverage values and layer thickness for each toner color on the paper.

To translate toner-coverage to color values, a spectral color prediction model is used [90]. This optical model is based on the combination of ray-tracing, Mie scattering and Monte Carlo simulations. It is applied to determine the color values of each pixel in the simulated print as a function of light source, toner ingredients, layer thickness, media type and toner area coverage. The model addresses effects as optical dot-gain, gloss and fluorescence. The output of the model is equivalent to an image that would be obtained by a perfect scan of the simulated print.

9.3 Virtual inkjet printer

Also for this model, first a description of the inkjet printing process will be given.

9.3.1 Inkjet printing technology

An inkjet printer contains a carriage that moves from left to right and from right to left over the paper. After each pass, a stepper motor moves the sheet of paper a certain distance. In this way, the carriage scans over each part of the sheet of paper. The carriage contains four printheads, one for each color (cyan, magenta, yellow and black). Each printhead contains an array of nozzles: the openings out of which drops can be jetted. These drops have a typical diameter of about 30 μm. An image is printed at a certain resolution of for example 600 dots per inch (dpi). This means that the image is divided into pixels of 42.3 μm (600 per inch). A drop can be jetted into each pixel. The drop size is fixed. Different shades of colors are printed by varying the amount of pixels in which a drop is jetted. Each pixel can contain more than one drop, but maximally one drop of each color. Image processing algorithms determine which colors are printed in each pixel. Due to variations in the jet speed or jet angle, drops can reach the paper at not exactly the right place. Drop placement errors and drop size variations determine the final print quality to a large extend. The shape of a dot on paper depends on the type of paper, the type of ink and the presence of other drops in the neighborhood.

9.3.2 Inkjet printer model

This model consists of four main modules as illustrated in Figure 9.9: image processing, drop positioning, fuse and the optical model. Also here the digital original image can be any full-color file on a PC and the image processing is exactly the same as in real printers. As before, the four different modules of the model can be changed or replaced independently. The final output of the model is a bitmap with color values for each pixel.

The image processing does the color management for the printer and calculates an optimal dot pattern for the printed image. Based on these desired drop positions, the movement of the paper and the movement of the carriage, the image processing generates a file which contains the moments each nozzle should jet a drop.

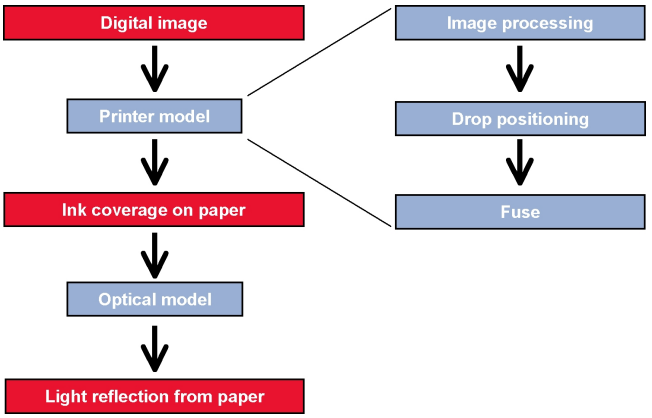


Figure 9.9: Flowchart of the inkjet printer model showing the different modules

The drop positioning module simulates the exact movement of the paper and the carriage, and calculates a flight path for each drop when it is jetted. The model includes paper step errors, printhead alignment errors and nozzle jet errors (jet timing, jet angle, jet speed, complete nozzle failure), which can all lead to drop positioning errors.

The fuse module simulates the ink coverage on paper, based on the drop positions that were calculated by the drop positioning module and the specified dot shape. Examples of different dot shapes are shown in Figure 9.10. In this module also interactions between drops can be specified. These interactions can cause drops to coalesce, which can have a severe impact on the print quality.

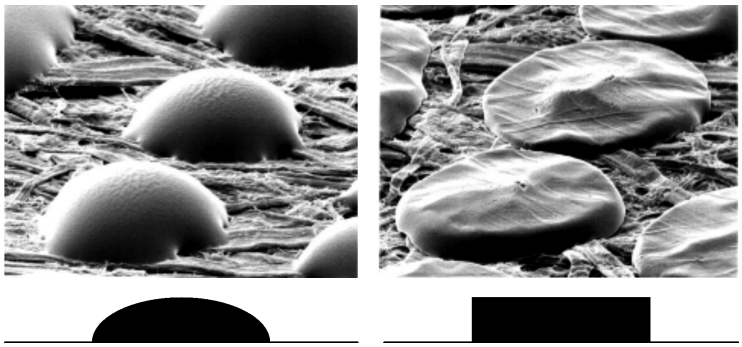


Figure 9.10: Electron microscope images of different dot shapes of ink on paper and the simulated dot shapes

The optical model simulates the interaction of light with the ink and paper. It includes optical effects such as surface reflection, absorption and scattering of light in the ink and paper, lateral light diffusion in the paper, optical dot gain and fluorescence

of the paper. The optical model calculates an XYZ color value for each pixel in the simulated print as a function of light source, ink reflection and scatter spectra, layer thickness, media type and ink coverage. Also for this model the output is equivalent to an image that would be obtained by a perfect scan of the simulated print.

9.4 Using the models

In this section two examples will be given of research that has been carried out with the virtual printer models.

9.4.1 Addition of grey toner

It was supposed by product developers that the addition of an extra DI-unit with grey toner in the DIP printing process should increase the print quality. The contrast between grey toner and white paper is less than between black toner and white paper, which should result in less visible noise (color variations with a high spatial frequency). Also more shades of grey can be made, which increases detail visibility. To investigate this hypothesis, the addition of an extra DI-unit with grey toner was simulated with the virtual DIP printer model. Also the image processing was changed in order to make use of the extra toner color. The results of the simulation are shown in Figure 9.11.

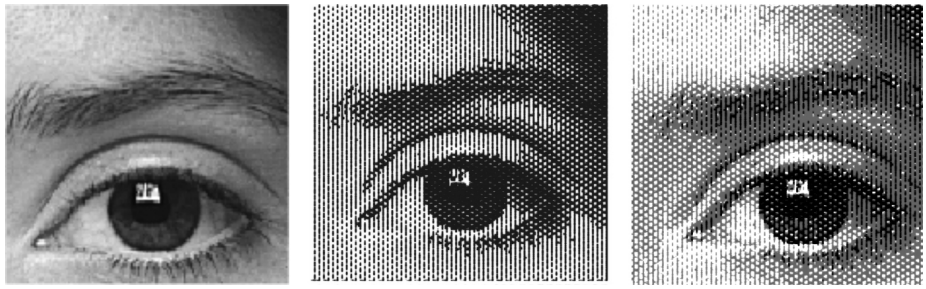


Figure 9.11: Part of an image (left) printed with only black toner (middle) and with grey and black toner (right). The real size of the images is approximately 8 mm

From the images it can be observed that the addition of an extra DI-unit with grey toner does indeed result in less visible noise and a better visibility of details. However, some contouring appears at certain shades of grey, which might be reduced by improving the image processing algorithms. Another drawback of adding an extra DI-unit are the extra costs which make the printer more expensive.

The virtual DIP printer model was able to predict the increase in print quality of adding a DI-unit with grey toner, and show the need for improved image processing. This prevented the need of having to build a printer with eight DI-units instead of seven

and the manufacturing of a special grey toner, which would have cost significantly more effort.

9.4.2 Resolution and number of drop sizes of an inkjet printer

For an inkjet printer that was to be developed at Océ, it had to be decided what the resolution should be. This printer would use printheads that were able to jet up to three different drop sizes from each nozzle. An experiment was performed to see how the print quality depended on the resolution and the number of drop sizes for this printer. Printed images with four different resolutions and different numbers of drop sizes were simulated. In Figure 9.12 a part of two of these images is shown.

Figure 9.12: Part of an image printed at 400 dpi with one drop size (left) and at 800 dpi with three drop sizes (right). The real image size is approximately 8 mm

A psychophysical experiment was carried out in which observers were asked to rate the print quality aspect ‘detail visibility’ of each of the simulated images. The setup and data processing of the experiment were carried out in the way that is prescribed in [38]. The results of the experiment are shown in Figure 9.13. The results show that with the use of two drop sizes the print quality increases significantly compared to the use of only one drop size. The use of a third drop size increases the print quality only slightly, especially at resolutions above 600 dpi. At resolutions higher than 600 dpi the print quality does not improve significantly, while the print speed would decrease due to the extra pixels that have to be addressed. Based on these results a resolution of 600 dpi with two drop sizes was chosen for the printer.

In this case the virtual inkjet printer model was used to derive printer specifications for an inkjet printer in an early phase of development. These results would otherwise only have been obtained when the development of the printer would have been almost

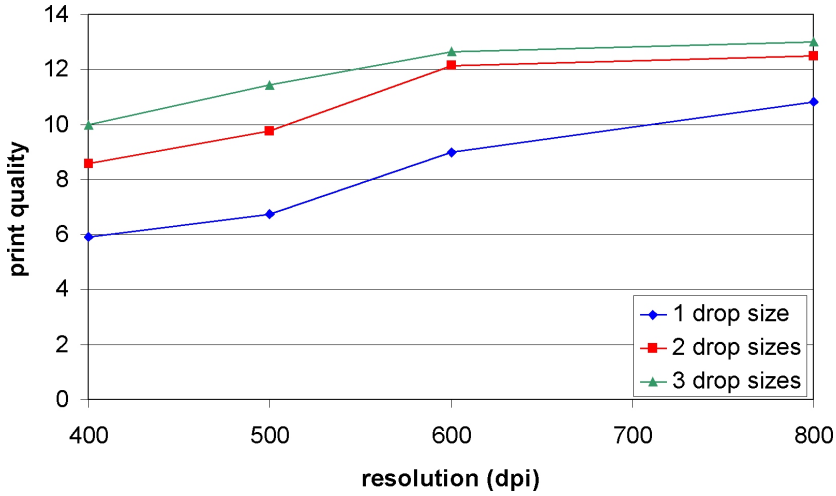


Figure 9.13: Rated print quality (detail visibility) as function of printer resolution and number of drop sizes

completed. By that time many (irreversible) design choices would already have been made.

9.5 Conclusions

Virtual printer models have been developed which can be used to simulate printed images of Océ's two color printing technologies (DIP and inkjet). The output of the models is an image that is equivalent to an image that would be obtained by a perfect scan of the simulated print. The virtual printer models consist of modules, which each represent a process part of a printer. The modules can be improved or replaced independently. Experts of each process part have contributed to the development of each module. By developing the models in such a multi-disciplinary way, the virtual printer models have become a valuable developing tool, which can simulate the effect of the properties of a printer on the final print quality with a high degree of accuracy.

The virtual printer models are used to:

- gain a better understanding of the impact of technical variables of a printer on the resulting print quality
- predict the potential print quality improvement of printer concepts that would cost considerable effort to design, manufacture and test in reality.
- derive design specifications for (parts of) printers in an early phase of the development process.

The usefulness of the models for these purposes has been demonstrated for various design cases, such as:

- the increase in print quality that can be achieved with the use of an extra DI-unit with grey toner
- the impact of the resolution and the number of drop sizes on the print quality of an inkjet printer.

Chapter 10

Using stepper motors in printers

Authors: J. Stolte, A. Veltman, P.P.J. van den Bosch and E.H. van de Waal

10.1 Introduction

In printers stepper motors can serve as a variable speed actuator for pinchers. Traditionally DC-motors with a rotational encoder are used because these motors are powerful and very easy to control. Due to their low total system cost stepper motors pose an interesting option to replace the DC-motors, even in spite of some disadvantages.

The dynamic behavior of stepper motors was unclear and confusing at the moment that the decision was taken to consider stepper motors as serious alternative. The fact that they can suffer from badly damped oscillations and even instability was acknowledged, but not understood. During the project this has been investigated, and this chapter shortly deals with the various topics designers should consider when implementing stepper motors as an actuator in any system.

10.2 Stepper motor (dis)advantages

Using stepper motor brings a number of advantages.

- Stepper motors are inexpensive because no incremental encoders are needed to track the rotor angle. For small motors the cost of an accurate encoder is greater than the cost of the motor itself such that savings are significant.

- In case a simple controlling scheme is used, the hardware needed to control stepper motors is both simple and inexpensive. In this case a feedback loop is not present.

- Stepper motors are very robust. They are not equipped with brushes that suffer from wear. The motor is not damaged if it stalls since the current will not increase dramatically.

Although the advantages of stepper motors are clear, they also suffer from some disadvantages. The most important ones for this application are:

- When driven in open-loop stepper motors are badly damped. Dynamic oscillations take a long time to decay, and for some controllers the motor will even be unstable at certain frequencies. This will be discussed in Section 10.5.2.

- The number of rotor teeth is typically large to increase the angular resolution. Disadvantage of having a high number of rotor teeth is that for higher rotational frequencies the motor produces a lot of emf voltage which needs to be overcome by the driver.

- Stepper motors have a low efficiency (typically 10-15%). Since stepper motors usually have a large phase resistance, large ohmic losses occur.

10.3 Stepper motor types

There are three main stepper motor types. Firstly, the variable reluctance motor uses a toothed rotor made of iron. Due to this rotor shape the magnetic reluctance varies with the rotor's angular position, and when a field is applied the rotor will try to settle in a position of minimal reluctance.

The permanent magnet type uses a magnet as rotor. The magnet generates a magnetic field which interacts with the current in the phase coils due to Lorentz' law. This interaction means the rotor will try to settle the rotor in a position where the fields are aligned. This type is essentially the same as a synchronous motor with fixed rotor current, but will generally have more pole pairs.

The third motor type is the hybrid stepper motor, which is a combination of the variable reluctance type and the permanent magnet type. It uses a toothed magnet rotor, which produces torque in the same way as the permanent magnet type, but has toothed stator and rotor to decrease the step size. In this way it combines the desirable aspects of both motor types. The hybrid motor is by far the most common motor type, and therefore this book will only consider the hybrid stepper motor.

Permanent magnet and hybrid type stepper motors have two options for their wiring schemes. Firstly there are the bipolar wound motors, which require the current through the coils to reverse polarity to reverse the field polarity. Secondly there are unipolar motors in which each phase actually consists of a pair of coils which are wound in opposite directions. In this case the field can be reversed by using the same current polarity through the other coil. The advantage is that less complicated hardware is needed as only positive currents are required. Disadvantage is that only half the coil volume is effectively in use at any given time.

10.4 Stepper motor modeling

Stepper motor phases are generally modeled as an inductance with a series resistance. This basic model is shown in Figure 10.1. The basic model does not cover effects like magnetic saturation, eddy currents and phase cross linkage. Since the model captures the modern hybrid stepper motor dynamics fairly well, there is no need to include these effects into the model. A notable exception is the skin effect, which will be discussed in Subsection 10.4.1

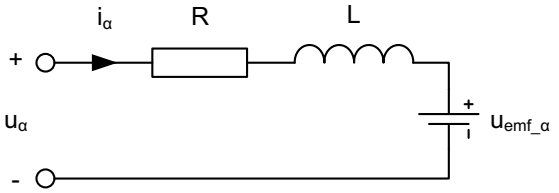


Figure 10.1: Electrical equivalent model of one phase of a stepper motor. The basic model only includes phase resistance, inductance and emf voltage generated by the motor. There are no cross-phase effects modeled.

The equations governing the basic model are given as equation (10.1) - (10.4):

$$L_\alpha \frac{di_\alpha}{dt} = u_\alpha - R_\alpha i_\alpha + p\lambda\omega \sin(p\theta) \tag{10.1}$$

$$L_\beta \frac{di_\beta}{dt} = u_\beta - R_\beta i_\beta - p\lambda\omega \cos(p\theta) \tag{10.2}$$

$$J\frac{d\omega}{dt} = -p\lambda i_\alpha \sin(p\theta) + p\lambda i_\beta \cos(p\theta) - B\omega - T_{load} \tag{10.3}$$

$$\frac{d\theta}{dt} = \omega \tag{10.4}$$

where i [A] is the phase current, u [V] is the phase voltage, ω [rad/s] is the rotor angular speed and θ [rad] is the rotor angle. The specific motor parameters are given by R [Ω] for phase resistance, L [H] for phase inductance, J [kgm^2] for total rotating inertia, p [-] for the number of rotor teeth, λ [Nm/A] for magnet strength and B [Nms/rad] for viscous friction parameter. The motor can also be subjected to a load and/or disturbance torque which is represented by T_{load} [Nm].

Note that the number of rotor teeth p actually acts as kind of internal gearbox which increases both the emf voltage and the torque. This implies the number of rotor teeth can also be viewed as scaling the motor constant. To produce one full mechanical revolution of the rotor the phase voltages and currents make p cycles. It is therefore useful to distinguish between the electrical frequency/angle ω_e, θ_e and the mechanical frequency/angle ω_m, θ_m. These are interrelated by $\omega_e = p\omega_m$ and $\theta_e = p\theta_m$.

10.4.1 Skin Effect

The skin effect is the most important non-modeled phenomenon. The skin effect dictates that for increasing frequency the current is pushed to the outside of a conductor. Even within the normal operating range of the motor the skin effect causes a significant increase in the phase resistance, and a significant decrease in phase inductance. For the NMB K404 of the Minebea Corporation a graph of phase resistance and inductance is shown in Figure 10.2.

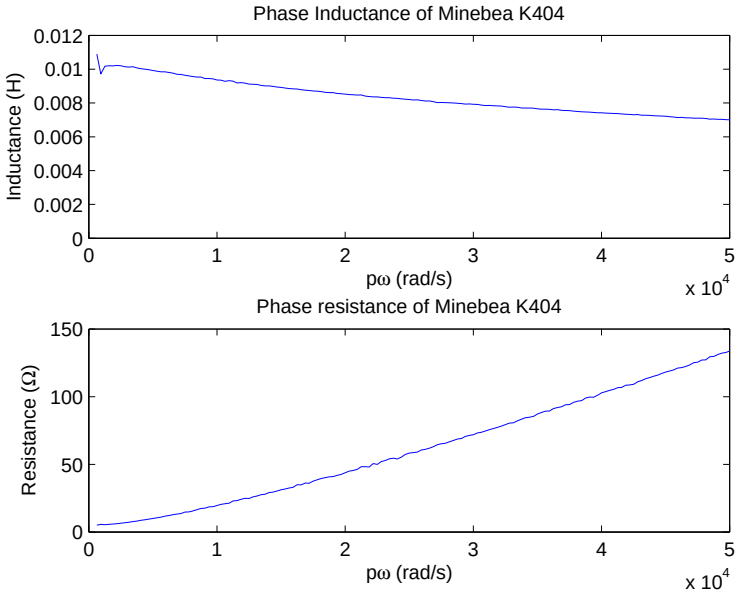


Figure 10.2: Results of skin effect in the Minebea K404 hybrid stepper motor on phase inductance and resistance. The time constant L/R becomes roughly 25 times smaller.

If the stepper model needs to be quantitatively accurate for a high frequency range the skin effect needs to be included into the model. This can be achieved by making the phase resistance and inductance parameters a function of frequency. Including the skin effect only serves model accuracy, and makes no fundamental difference in the modeling of the motor. Because models including skin effect become much less transparent the models in this chapter do not include skin effect, but could be easily adapted for it.

10.4.2 IRTF simulation model

In many analyses of stepper motors some kind of coordinate transformation is applied such that the electrical quantities are viewed in a rotating system. The voltages and currents in the stepper motor phases can be seen as vectors, which rotate along with

the rotor. During steady state rotation the voltage and current vectors can be viewed as constant, which greatly simplifies analysis.

A new stepper motor model is proposed, based on the Ideal Rotating TransFormer (IRTF) concept presented in [116]. The IRTF models the stator circuit, the rotor circuit and the motor mechanics separately, which means it has an inherent coordinate transformation. Since the stator consists of two orthogonal phases the stator flux/current can be represented as a vector in two dimensions. The rotor is represented by a fixed permanent magnet which is rotated over the rotor angle. The IRTF is shown in Figure 10.3, and the equations governing it are given by:

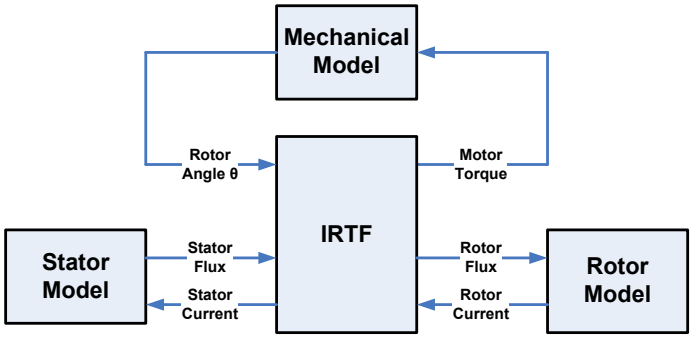


Figure 10.3: IRTF block model. The IRTF realizes a coordinate transformation between the rotor and stator coordinate systems. Motor current and flux are represented as vectors, which are rotated back and forth over the rotor angle by the IRTF.

$$\vec{\Psi}^R = e^{-jp\theta}\vec{\Psi}_S \tag{10.5}$$

$$\vec{I}^S = e^{jp\theta}\vec{I}^R \tag{10.6}$$

$$T = \vec{I}^R \times \vec{\Psi}^R = -\vec{I}^S \times \vec{\Psi}^S \tag{10.7}$$

where the $\vec{\Psi}^R$ represent the magnetic flux in the system represented as a vector. The superscript R means the vector is defined in the rotor coordinate system, while superscript S refers to the stator coordinate system. T is the torque produced by the motor.

The IRTF concept can be used to model any electric machine. The stepper motor model based on the IRTF is elucidated in Figure 10.4. It can be shown that the IRTF stepper motor model is mathematically equivalent to the basic model presented as (10.1)-(10.4), It is merely a novel, more clear implementation of the basic model.

Each implementation has it's own specific advantages. We prefer a model based on the IRTF concept for several reasons:

- The IRTF model is close to physical reality. Apart from the phase voltages and

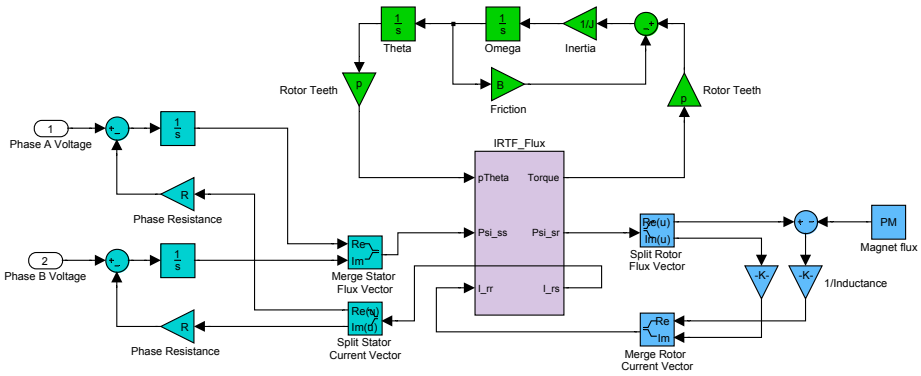


Figure 10.4: Simulink implementation of the IRTF stepper motor model.

currents, the IRTF model also includes the magnetic flux in the motor's phases. All these quantities can be individually measured and influenced.

- Detent torque, skin effect, saturation or phase cross-coupling are relatively easy to include due to the explicit availability of all stepper motor variables.

- The model is flexible in handling all kinds of scenarios. It remains valid for all wave shapes and rotor positions/speeds. Stalling behavior for instance is easily and accurately modeled.

10.5 Stepper motor control

Driving stepper motors can be done in various ways, which are all slight variations of the same principle. Since it is economically attractive to drive stepper motors without the use of an encoder, they are usually driven in open loop. However, the dynamic behavior of stepper motors in open loop is poor, due to the inherently low damping in the system. In closed loop stepper motors can be actively damped, but this requires knowledge of the rotor angular position/speed. This knowledge can be obtained through the use of an encoder (expensive hardware), or with an observer based on measuring the emf voltage produced by the motor (high software load).

In this section the general idea of driving stepper motors is demonstrated. Next, various ways of implementing stepper motor drivers are given in order of increasing complexity (and performance).

10.5.1 General stepper motor driving

Stepper motors are offered (quantized) rotating vectors of voltage or current. The motor will rotate to match the angle of the input vector. For a hybrid stepper motor which is

subjected to a static current, the behaviour is drawn schematically in Figure 10.5. In this picture the motor is shown to have only one pole pair ($p = 1$) for sake of simplicity.

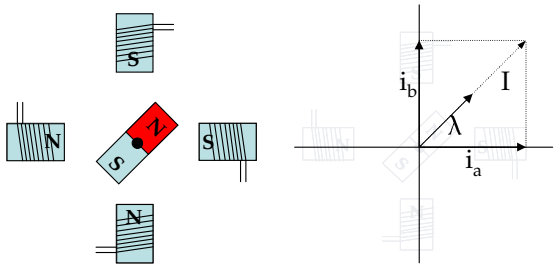


Figure 10.5: Schematic representation of a stepper motor with both phases excited. The phase currents i_a and i_b create a magnetic field in the stator. When taken together they can be viewed as the current vector I. The permanent magnet generates it's own magnetic field, which is represented by the flux vector λ. In the drawn situation the flux and current vectors are aligned, which means that no torque is generated and the system is in rest.

If one of the phases in Figure 10.5 is excited with reversed polarity, the input vector is rotated over 90 degrees. Due to this rotation there is now a mismatch between the alignments of the magnetic field generated by the current and the magnetic field generated by the permanent magnet in the rotor. This mismatch causes a torque which in turn causes the rotor to start rotating in the direction of good alignment. When the rotor is settled again the motor is said to have made one full step. A typical rotor movement when a full step is applied to the input is shown in Figure 10.6. Sequentially making full-steps in the same direction will cause the stepper motor to keep rotating and this way of driving stepper motors is called full-stepping.

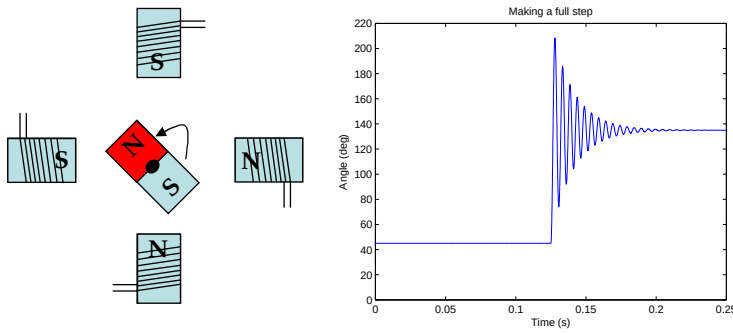


Figure 10.6: Making a full step. The horizontal phase is reversed in polarity, which causes a 90 degree rotation. In the right figure the rotor angle is shown as a function of time.

Note that the voltage waveforms offered to the motor phases in full-stepping mode are square waves. The waveforms of the two phases are identical but shifted over a quarter of a period. These square waves can be viewed as a quantized version of sinusoidal waves. Since the overshoot and settling time of one step are proportional to the step angle it is beneficial for the dynamic behavior of stepper motors to make smaller steps. Figure 10.7 shows the input signals for half-stepping[1] and 16-micro-stepping, which is in essence corresponds to sampling the sinusoidal signals more often.

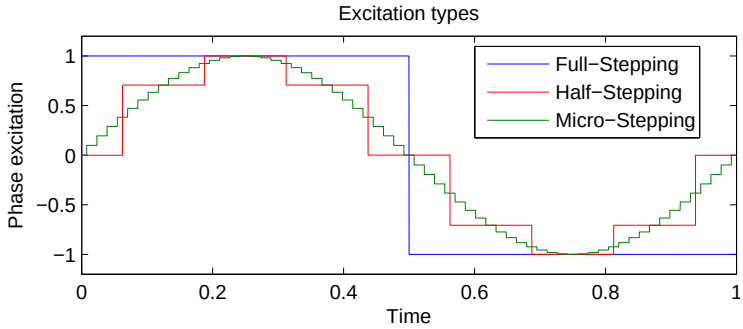


Figure 10.7: Phase excitation waveform for different excitation options.

If smooth rotation is desired, it can be profitable to make even smaller steps. This practice is called microstepping, and drivers dividing full-steps into up to 256 microsteps are widely available. Using a PWM generator to directly control the phase voltages/currents, even smoother waveforms can be made.

10.5.2 Open loop voltage control

The most inexpensive way of controlling the stepper motor is through open loop voltage control. No measurement of any kind is needed, just put the revolving voltage vector according to the stepping method directly onto the phases. If the voltage amplitude is increased for higher frequencies, the steady state current can be kept approximately equal to the motor's nominal current. For varying load torques, the actual steady state currents will differ slightly from the nominal case.

The stability of this control scheme has been studied analytically through linearisation and inspection of the system's four eigenvalues. This analysis revealed in this scheme oscillations are not only damped by the viscous friction, but also by the change in current caused by the emf voltage produced by the motor. The downside is that the additional damping depends on frequency, and for higher frequencies it actually becomes negative which causes the system to become unstable [63].

[1] Note that in practice half-stepping is often implemented with only $+1, -1$ and 0 phase excitation. In this case the hardware is simpler, but the motor torque will not be constant.

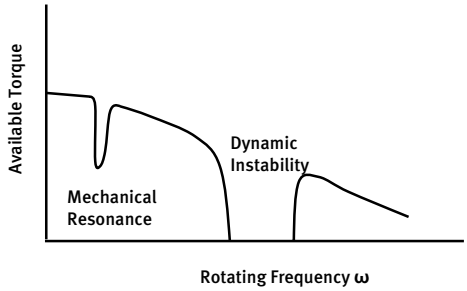


Figure 10.8: Schematic representation of available torque as a function of frequency, when using open-loop voltage control.

In Figure 10.8 the practical dynamic instability region is shown. The frequency at which this instability actually occurs is lower-bound by $p\omega = R/L$. For low total rotating inertia, the frequency of instability will actually be significantly higher [109]. There is another dip in the available torque, which corresponds to the motor's mechanical resonance frequency which is discussed below.

10.5.3 Open loop current control

Only slightly more difficult to implement is open loop current control. In this scheme the motor is offered a (quantized) rotating current vector instead of a voltage vector. Since the available torque is directly proportional to the phase current, this method is often preferred.

It is important to realize that, since stepper motor phases are inductive loads, true current control is never possible because inductances do not allow for discontinuities in the current. In practice voltage-choppers are used to control the phase currents. At higher frequencies, the voltage available will not be sufficient to produce the desired current. If this is the case, the stepper motor controller essentially becomes an open loop voltage controller as described in Subsection 10.5.2.

Because the current controller forces a certain current through the phases the differential equation for current of the stepper motor model becomes irrelevant. As a result the fourth-order system is reduced to a second order mechanical system. A side effect is that there is now no influence of the emf-voltage generated by the motor on the motor dynamics. Therefore the emf-voltage does not generate any additional damping term like with open loop voltage control. As a result the system will not become unstable, but will also not have the beneficial damping effect at lower frequencies.

The damping of the resulting second order system can be found by looking at the system poles, which are given in (10.8). The negative real value of the system poles is directly proportional to the viscous friction coefficient B. Therefore having a low friction parameter is very bad for damping of oscillations in the system, even though it would be desirable for high speed performance and efficiency considerations. As-

suming that $B^2 \approx 0$, the part under the square root gives rise to a natural frequency, which is dependent on the load torque. In most practical situations this dependence can be neglected as well and the mechanical resonance frequency can be approximated by $f_n \approx \frac{1}{2\pi}\sqrt{\frac{p^2 \lambda I}{J}}$.

$$s = \frac{-B \pm \sqrt{B^2 - 4pJ\sqrt{(p\lambda I)^2 - T_{load}^2}}}{2J} \tag{10.8}$$

Because the motor is very badly damped at the natural frequency, exciting this frequency too much should be avoided. When the motor is driven at or near it's natural frequency the motor will quickly lose synchronization and stall. This dip in available torque is the same as the mechanical resonance in Figure 10.8 except it is worse for current control. However, for current control the dynamic instability region is gone.

Due to the wide use of this control scheme, a number of workarounds for mechanical resonance have been devised. They include the use of inertial dampers, additional electronics and flexible mounting rings. All of these have in common that they selectively dissipate energy from oscillations at or around the mechanical resonance frequency.

10.5.4 Closed loop control

So far only open loop control options have been considered. These schemes have in common that they force some revolving input signal on the phases, and it is assumed that the rotor revolves in synch with the input. The main problem with the open loop concept is that the motor can be badly damped, or even become unstable. Should the stepper motor lose synchronization, the controller will not even notice it which is quite undesirable.

In contrast, if the rotor angular position/speed are known to the controller it can use this information to improve control performance. By slightly changing the angle or amplitude of the applied control vector the controller can stabilize the system if it is unstable, and greatly improve the damping.

Much effort has gone into closed loop control of stepper motors, which has lead to various controller designs [126, 34, 1]. Recently, a proposal was made by Yang [126] for active damping current control based on an observer which showed excellent results. The simulated effect of adding a controller on system damping is shown in Figure 10.9.

Of course, adding an observer and a controller implies additional software and hardware costs when compared to the open loop control options.

10.5.5 Observer

To close the control loop the stepper motor needs to be aware of the rotor angle/speed. Classically this information would be gathered from a mechanical sensor. But since

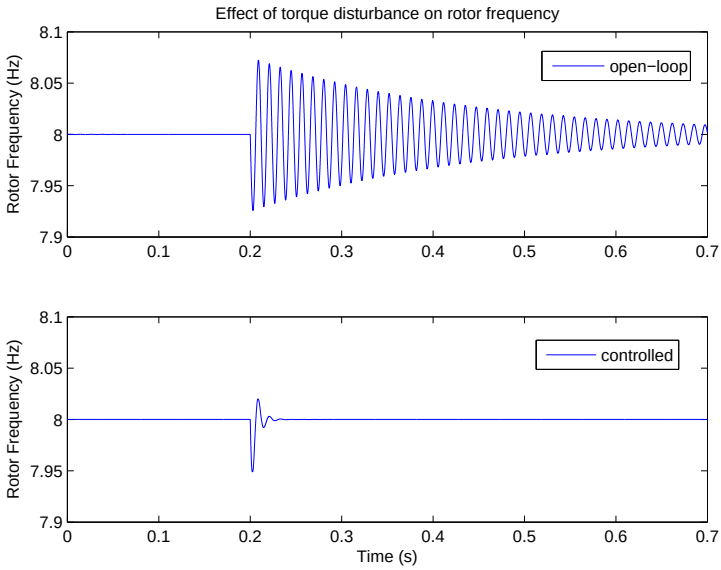


Figure 10.9: Simulated effects of a small torque disturbance on the stepper motor speed. Open-loop the system is badly damped, which can be improved by adding a controller.

using an encoder significantly increases the total system cost, other means of gaining this information have been developed using the emf voltage. By measuring the phase voltages and currents the emf voltage generated by the motor can be reconstructed [126, 104].

Looking at the differential equations for the phase currents (10.1) and (10.2) it can be seen the emf-vector holds information about the rotor angle (in the phase), and about the rotor speed (in the amplitude). By tracking the emf-vector an accurate estimation of the rotor position and speed can be made under the condition that the rotor speed does not change too fast and the speed is not too low.

The magnitude of the emf-vector is directly proportional to the rotational speed of the rotor. That implies that at low speeds the motor it is difficult to reconstruct the actual rotor angle and speed. Fortunately the worst behavior of instability for voltage control, and the resonance frequency for current control generally occur at higher frequencies.

Another downside is that reconstructing the emf-voltage and computing an angle/speed estimation means the computational demand of the system will rise significantly.

10.6 Conclusions

Hybrid stepper motors are suited as low cost actuators. The low cost and high robustness make them an interesting option. Some care needs to be taken with what motor to take, and with what control concept to drive it. Although very closely related each method has its own points of attention.

Except for the skin effect, the tested stepper motors can be accurately modeled using a simple model. For simulation purposes a model has been developed based on the Ideal Rotating TransFormer (IRTF) concept. The most important advantages of the model are good visibility of system variables, and high flexibility in implementing different control scenarios.

There are several options available for implementation of stepper motor control. Firstly, the size of the steps offered to the motor can be varied. Smaller step size requires more complicated hardware, but offers smoother rotation. Secondly, the control concept needs to be chosen. Open-loop voltage control is very easy to implement, but leads to instability at high frequencies. Open-loop current control does not suffer from instability but causes a badly damped natural frequency in the system. Closed loop control solves the dynamic problems but is more expensive to implement as it needs knowledge about rotor angle/speed.

It is to be expected that with the proper control stepper motors will replace DC-motors in printers with preservation of performance.

Chapter 11

Simulating the environment of embedded software

Author: J.J.M. Hooman

11.1 Introduction

The development of mechatronic systems requires the involvement of several disciplines, such as electrical engineering, mechanical engineering, and software engineering. Although these disciplines are tightly coupled in the considered mechatronic systems, their development was traditionally a rather sequential, mono-disciplinary, process. Typically, first the mechanical part was designed, next the hardware infrastructure was fixed, and finally the embedded software had to be developed.

This approach can create large problems, especially for the software engineers. For instance, choices about the placements of sensors (and implicitly the timing of interrupts), control rates, control delays, execution platform, et cetera, have a strong influence on the complexity of the software. Moreover, usually many implicit assumptions are made, which first become visible at system integration. This easily leads to non-optimal solutions and long development cycles.

To improve this situation, aiming at a shorter time-to-market, there is a clear trend towards concurrent engineering where disciplines work in parallel and, for instance, software is developed before the mechanical part has been finalized completely. This, however, requires techniques to test embedded software before its environment is ready. Moreover, it is important to synchronize design decisions and identify problems during early stages of multi-disciplinary development.

To detect problems as early as possible, all disciplines make heavy use of models. Moreover, mono-disciplinary modeling is often supported by tools that allow some

form of execution or simulation. Lacking, however, are methods which combine models and tools of different disciplines. This chapter addresses the question whether models can be used to simulate and to test embedded software in combination with a simulation of its environment. We describe three related research activities within the Boderc project and the application in the context of Océ.

The software at Océ is modeled using the UML-based CASE tool Rose RealTime (currently renamed to Rose Technical Developer) of IBM Rational [64]. Rose RealTime, henceforth called RoseRT, supports the ROOM methodology [105] for the software development of real-time reactive systems. The tool allows the generation of code for a particular target platform.

The Matlab/Simulink [106] tool of The Mathworks is used at Océ to model the mechanical layout of a printer and to experiment with, for instance, the shape and the length of the paper path, the placement of motors and sensors, and the paper speed. A successful example is the so-called Happy Flow model, as described in Chapter 6.

Given the current tool support at Océ, the focus of this chapter is on modeling in RoseRT and Simulink. These tools are presented briefly in Sections 11.2 and 11.3, respectively. Next, we discuss, in chronological order, three approaches that have been investigated within Boderc to simulate embedded software in its environment:

- Section 11.4 presents the coupling of the tools RoseRT and Simulink, to obtain simultaneous simulation – also called co-simulation – of models from different disciplines. This research has been performed by Nataliya Mulyar, Ladislau Posta and Jozef Hooman.

- Section 11.5 discusses a framework where a specific part of the Océ software (viz., the real-time control part which has to be executed in a periodic way) is integrated into Simulink using the Simulink toolbox TrueTime. This framework has been developed by Zhaorui Yuan and Peter van den Bosch.

- Section 11.6 adapts the framework of Section 11.5 by replacing Simulink by a simulation in C++ which has been developed by means of RoseRT. This work has been carried out by Sebastiaan van der Hoest and Lou Somers.

Finally, Section 11.7 contains concluding remarks.

11.2 RoseRT

RoseRT is a UML-based CASE tool for the development of complex reactive software. Typically, a UML model in RoseRT consists of a number of active objects, also called capsules, which communicate by sending and receiving messages via ports. Messages may have different priorities. The behavior of a capsule is modeled by means of subcapsules or a state diagram. Transitions in a state diagram are triggered by the reception of messages or time-outs. Actions on a transition may change local variables, send messages, or set timers.

Given a complete model, the RoseRT tool can generate code, based on a so-called Service Layer which provides an abstraction of the underlying execution platform. The Service Layer provides general services, e.g. it contains controllers, which are responsible for queuing and delivering messages among capsules. The timing service of the Service Layer provides the model developers with general-purpose timing facilities based on both absolute and relative time. The implementation of the Service Layer depends on the target platform on which the program should run. Hence, the precision of the timing service depends on the granularity of the timing supported by the underlying operating system.

RoseRT supports two ways to validate its UML models. The first possibility is to execute the model, i.e. the generated code. The second way is to execute the model step-by-step, where a step is associated with processing the next message of the highest available priority. A step terminates when all actions that are a consequence of this message have been performed. In the second way, timers are running independently from the steps, and – compared to the first way – time-out events may occur at different moments in an execution trace. In general, model validation is mainly intended to check the response to events and is not suitable to guarantee timing behavior. However, because of time-outs, the reactive behavior of a system may also depend on timing.

11.3 Simulink

A Simulink model is represented graphically by means of a number of interconnected blocks. Lines between blocks connect block outputs to block inputs. Blocks may have states, which may consist of a discrete and a continuous part. The output of a block is computed by an output function, based on its input and its current state and time. Similarly, an update function calculates the next discrete state. A derivative function relates the derivatives of the continuous part of the state to time, the current values of the inputs, and the state.

During the simulation of a Simulink model, the outputs, inputs and states are computed at certain time points. The successive states of a system are computed by a so-called solver, a Simulink-specific program. Since no solver is suitable for all models, there are several types of solvers. The solvers use numerical integration to compute the continuous states of a system from the state derivatives specified by the model. Each solver uses a different integration method, allowing the selection of the most suitable method for a particular model. The amount of time between successive time points at which the states and outputs are computed is called the step size. This step size depends on the type of the solver used, the characteristics of the Simulink model, the accuracy required, and the existence of discontinuities in the model.

11.4 Coupling RoseRT and Simulink

The general aim of the work described in this section is to support multi-disciplinary development by combining modeling tools from different disciplines. Given the Océ context, we have implemented a coupling between RoseRT and Simulink. As an example, our coupling allows the combination of a continuous model of a physical dynamical system in Simulink with an event-driven control algorithm in RoseRT, as depicted in Figure 11.1.

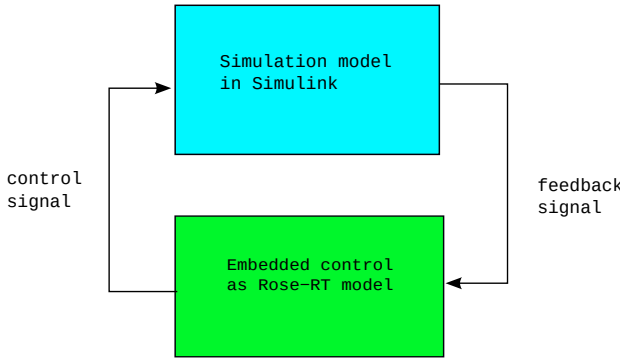


Figure 11.1: Example of the combination of models

By establishing a proper notion of co-simulation, one can for instance investigate the effect of changes in the control strategy or the mechanical lay-out on the software. Vice versa, it allows an analysis of the impact of software delays on global system behavior. Realizing the desired tool coupling is far from trivial. The two main challenges are: (1) Conceptual correctness; the coupling should be such that the simultaneous simulation of models in both tools gives meaningful results – in particular, there must be a common notion of simulation time. (2) Technical implementation; the coupling software should be properly designed to allow, for instance, a change to another UML tool without too much effort.

In Subsection 11.4.1 we give the most important design decisions of the tool coupling and the main concepts of the implementation. Subsection 11.4.2 contains an evaluation of the coupling. More details can be found in [60].

11.4.1 Design decisions of the coupling

Notion of time

The most important decision concerns the notion of time to be used in the simulation. Observe that the timing of RoseRT is strongly coupled to the timing service of the operating system of the platform on which the model is running. Moreover, as described in Subsection 11.2, timing is not respected in the step-by-step execution. Hence, we

concluded that RoseRT does not offer a proper notion of simulation time and decided to use the notion of simulated time of Simulink instead. The alternative is to use a separate, independent, notion of time, but this would also require new implementations of solvers, redoing a lot of work on functionality already available in Simulink.

Observe that actions on transitions in a RoseRT model may involve large computations and the computation time of transitions cannot always be neglected. For instance, when a new set point is computed, the computation delay may have an observable impact on the controlled object. Hence, the user has to specify the duration of transitions in the RoseRT model and these software delays have to be taken into account in the Simulink model. E.g. a set point computed in the RoseRT model should only be used in Simulink after a proper delay which reflects the computation time. Note that the specified durations of transitions express assumptions on the execution time of the generated code on the target platform.

Summarizing, the implementation of the tool coupling has to ensure that the timing service used by the RoseRT is obtained from the simulation time of Simulink. For instance, time-outs generated by timers in the RoseRT model should correspond to the simulated time in Simulink. Moreover, it has to be ensured that Simulink uses data from RoseRT after the specified software delay.

Global architecture of the coupling

Another decision taken is the global architecture of the coupling. Instead of a tight coupling, we decided to use a more loosely coupled architecture by introducing a third component called Multidisciplinary Coupling Tool (MCT), as shown in Figure 11.2. Observe that each tool contains an add-in, which is responsible for the communication with the MCT component.

By introducing such an MCT interface, the modeling tools do not need to know about each other and it becomes much easier to change one of the tools. For instance, to switch to another UML-based CASE tool. The MCT component consists of three interfaces: a Remote Control Interface, a Data Interface, and a Timing Interface.

- The Remote Control Interface allows starting, stopping and controlling the execution of the RoseRT model in step-by-step mode (arrow 4 in Figure 11.2). This functionality can be accessed by the MCT Simulink add-in (arrow 7).

- The Data Interface serves as a storage for data exchange between the RoseRT and Simulink models, including the timing delays associated with the execution of transitions in RoseRT (arrows 5 and 8).

- The Timing Interface keeps track of the simulation time. It represents an intermediate clock, which is updated with the value of the Simulink simulation time (arrow 9) and which is regularly sampled (before a step in RoseRT is executed) by the Timers of the Simulated Target Operating System (arrow 6).

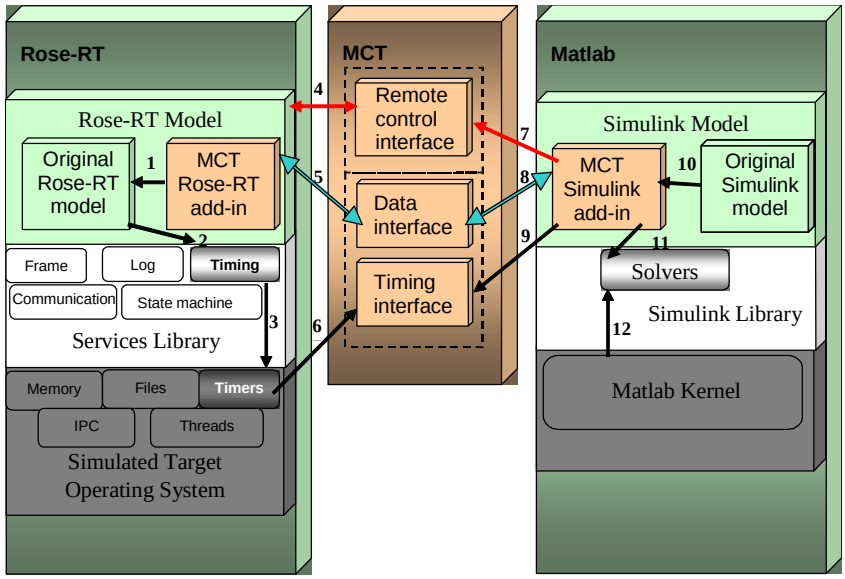


Figure 11.2: Combining models

11.4.2 Evaluation of the coupling

We realized a first prototype of a coupling between RoseRT and Simulink, which was to our knowledge the first coupling between a UML tool and Simulink. A few simplifications have been made to obtain a first prototype quickly. For instance, only one timer is allowed in the UML model. In principle, the approach is very generic and can be used for arbitrary applications. It nicely exploits the visualization possibilities of both tools, although step-wise simulation is rather slow.

Related is the work on the High Level Architecture (HLA) [35], a general-purpose architecture for the coupling of simulation tools. However, HLA cannot be used for our purpose, because the RoseRT tool does not include a simulation mode with a well-defined notion of simulation time, as required by the HLA framework. Recently, Telelogic [110] announced a combination of the UML tool Rhapsody and Simulink.

A disadvantage of our current approach is that the user has to specify the duration of transitions. These numbers are often not known and difficult to estimate, see for instance Chapter 8. Then a sensitivity analysis with respect to software delays can still be useful. A problem for the use at Océ is that there are no suitable Simulink models available that can be coupled to RoseRT models easily. For instance, the Happy Flow model of Chapter 6 abstracts from many details of motors and sensors, including the interface to the software.

11.5 Integrating software in Simulink using TrueTime

An alternative approach to couple embedded software and its environment has been realized using TrueTime [113], a Simulink toolbox. Similar to the previous section, the environment is simulated using Simulink and the software runs in the simulated time of Simulink. However, instead of simulating the RoseRT model, the generated C-code is used and simulated by means of TrueTime. With TrueTime it is easy to simulate multiple processors, and the toolbox includes simulated communication, interrupts and I/O functions.

Another important difference is that the goal is not to produce a generic coupling tool, but to test a specific part of the Océ software, namely the real-time control software for the paper path of a printer. This piece of software has to run in a strictly periodic way, reading sensors and controlling motors at a specific frequency and is scheduled in a time-sliced way. Hence, actions should have a duration shorter than the time slice, and for this application it is assumed that the duration of actions can be neglected, using duration zero.

11.5.1 Overview of the TrueTime approach

The general approach is depicted in Figure 11.3, showing the three main parts:

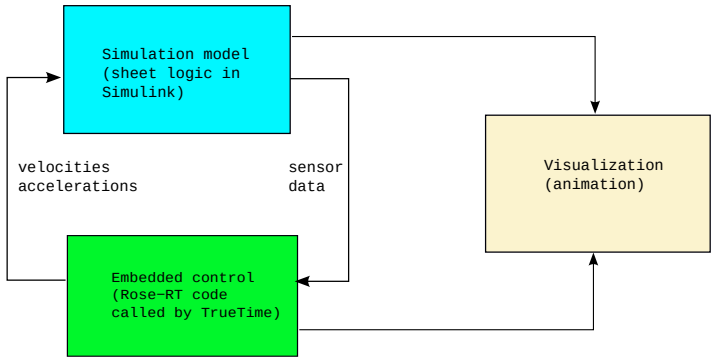


Figure 11.3: Testing framework based on TrueTime

- A Simulink ‘sheet logic’ model of the paper path lay-out, keeping track of the positions of papers as a function of time. Abstractions are made of the complete physical behavior of sheets. For instance, to improve simulation speed, motors are not modeled; it is assumed that the requested speed profile is directly present on the pinches.
- The control part, which is based on the code generated from a RoseRT model. This code is called by TrueTime with the requested frequency. The IO-layer with hardware drivers has been replaced by drivers that interact with the model.

- A visualization part which, for instance, animates the sheet movements in the paper path.

11.5.2 Evaluation of the TrueTime approach

The framework based on TrueTime has been used in a project at Océ. With this simulation framework, embedded software can be tested without physical lab prototypes, which enables parallel development of mechanical parts and software. It has been observed that the simulation framework indeed reduces the software development time. Moreover, it turns out that the framework can also be used for debugging and understanding lab model prototypes.

A disadvantage of the approach is that an optimal usage of the framework requires some Matlab/Simulink skills. E.g. when debugging an application, a lot of useful information can only be obtained by an experienced Matlab/Simulink user. Moreover, at the early stages of development (when this kind of tool support would be most useful), there are frequent changes in the lay-out of the paper path. This implies frequent updates of the Simulink 'sheet logic' model, which is difficult for software engineers and makes their progress dependent on the availability of Simulink experts. Note that this also holds for the approach of Section 11.4.

11.6 Replacing Simulink by a software-based simulation

To overcome the problem mentioned in Subsection 11.5.2, we consider an approach which aims at improving the usability and maintainability of the testing framework described above by software engineers. Therefore, Matlab/Simulink has been replaced by a software-based simulation which can be developed with the standard software engineering tool at Océ, namely RoseRT. Initially, this was done using capsules (active classes), but to increase the speed of the simulation, avoiding message passing overhead, these have been replaced by passive classes from which C++ code is generated. More details can be found in [59].

Similar to the TrueTime approach, the C++-based simulation has been developed to test the same hard real-time part of the Océ software. In the C++ simulation, a fixed time step is used, but the step size can be adapted depending on the required accuracy. Since the aim is to simulate interactions of the software with sensors and actuators, also motors are included. As in Section 11.5, execution times are neglected and assumed to be zero. The output of the simulation is visualized by means of an animation.

11.6.1 Evaluation of the software-based simulation

The software-based simulation environment has been used in an Océ project, where it became the default simulator for software testing. Based on feedback from Océ em-

ployees, a few features were added such as the possibility to run multiple test cases automatically. A disadvantage of the framework is that it is rather mono-disciplinary and there is a danger that the simulation environment diverges from the actual mechanical design. Hence, it is important to have a common configuration file for the paper path lay-out, including types of motors and sensors, which is used by all disciplines.

11.7 Concluding remarks

We have shown three approaches that simulate the environment of embedded software. The tool coupling of Section 11.4 provides a general framework to combine and simulate a RoseRT model with a Simulink model of its environment. Visualization is obtained by the graphical possibilities of both tools. Section 11.5 aims at testing a specific part of the Océ software, namely the embedded real-time control part with time-sliced scheduling. TrueTime has been used to embed the code generated by RoseRT in Simulink. Visualization has been obtained by dedicated animations of the paper flow. Application at Océ was successful and reduced the development time. To improve the usability and maintainability of the framework by software engineers, Simulink has been replaced by a software simulation in C++, as described described in Section 11.6.

Moving from Simulink to C++-based simulation increases the risk that models of different disciplines are inconsistent. On the other hand, experience at Océ shows that especially the animation possibilities increase multi-disciplinary communication and cooperation. It provides a common view which is useful to discuss observed problems and the consequences of changes.

Chapter 12

Evaluating embedded system architectures

Authors: M.H.G. Verhoef and J.J.M. Hooman

12.1 Introduction

In this chapter, we investigate several techniques that can be used to evaluate performance properties of embedded system architecture such as latency, throughput and resource utilization. We focus on these properties because they play a significant role in the selection of a suitable embedded architecture. The challenge is to decide, at design time, how to distribute functionality on a proposed embedded architecture, or, how to select suitable architecture parameters, such that required performance targets and cost levels are met. Typical questions that are raised at design time are: (i) does the architecture meet the performance requirements of all applications (ii) how robust is the architecture with respect to changes in application or architecture parameters and (iii) is it possible replace components in the architecture by cheaper, less powerful, components to save cost while maintaining the required performance targets? We will focus on the first question here, by applying four techniques to a case study. The aim of the experiment is to better understand the capabilities and limits of each method and to determine the value of the predictions derived from each model.

First, we describe a small case study that was inspired from industrial practice in Section 12.2. Then, in Sections 12.3 – 12.6, we introduce the modeling techniques. We present the results gained from the case study in Section 12.7 and we discuss the lessons learnt from our experiments.

12.2 The In-Car Radio Navigation case study

12.2.1 Description of the system

In this chapter, we study the design of an in-car radio navigation system. Such an infotainment system typically executes several concurrent software applications that share a common, and often distributed, hardware platform. We consider only a small subset of the in-car radio navigation system for our comparison, characterized by three clusters of functionality: the man-machine interface (MMI), navigation (NAV) and the radio (RAD). Three applications are executed by the system concurrently; *ChangeVolume*, *AddressLookup* and *ReceiveTMC*. Each application is independent and composed of functions provided by the three clusters mentioned earlier. Each application is described by a UML sequence diagram that is augmented with performance data. For example, the *ChangeVolume* application is presented in Figure 12.1. In addition, information is provided on the priority of tasks and messages and the scheduling methods used on each resource [1].

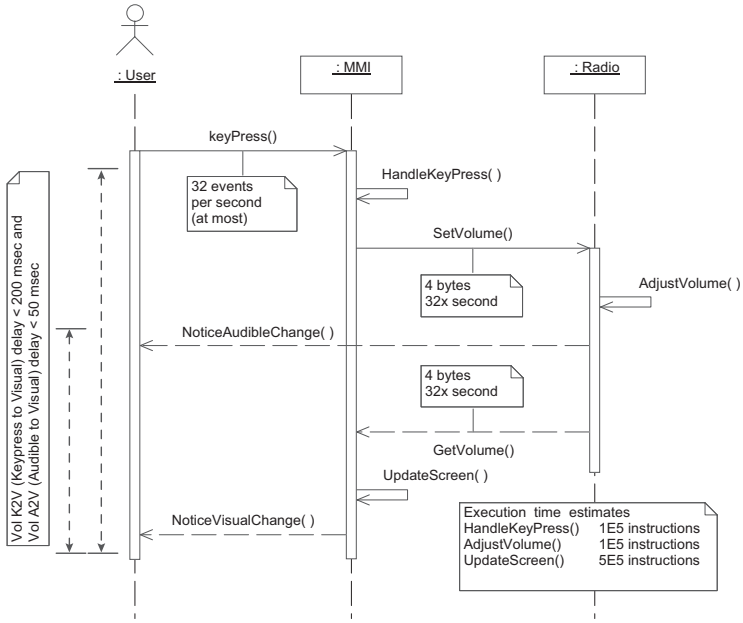


Figure 12.1: Augmented UML sequence diagram for “ChangeVolume”

Each application has individual requirements that need to be met and the question is whether *all* requirements can be satisfied when a particular architecture is chosen. Example deployment proposals are shown in Figure 12.2. We concentrate on Architec-

[1] Not shown here, a full description can be found at *http://people.ee.ethz.ch/∼leiden05/* .

ture (a) in the remainder of this chapter. We will investigate whether the combination *ChangeVolume* and *ReceiveTMC* and the combination *AddressLookup* and *Receive-TMC* meet the system-level requirements for this architecture.

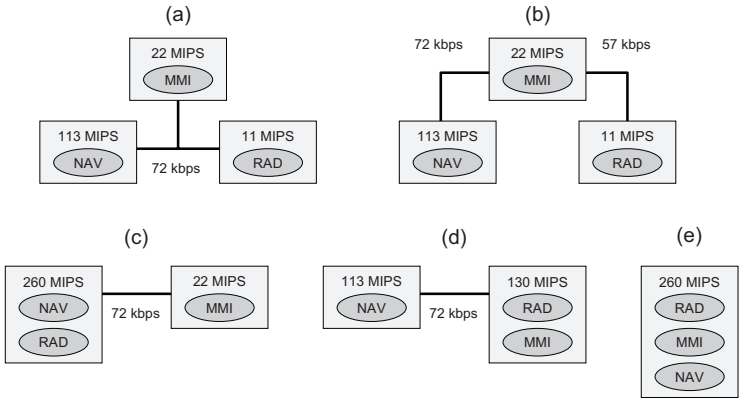


Figure 12.2: Alternative system architectures to explore

12.2.2 Environment of the system

In order to analyze the proposed embedded architecture, we also need to characterize the so-called *workload* that the environment imposes onto the system. In this case study, we simply describe how often each application is invoked. We can abstract away from the complexity of the environment (which might be another embedded system) by describing the stimuli as a (p, j, d, o)-tuple. The p parameter describes the period of the stimulus, j describes the jitter, d the minimal inter arrival time and o the offset for the start of the first period. Most stimuli arrival patterns can be described or approximated by this approach, including burst and sporadic behavior. The relationship between the parameters is graphically depicted in Figure 12.3. The (p, j, d, o)-tuple basically defines the *time interval* in which a stimulus will occur. This model can be enriched with a stochastic variable which defines the distribution of the event within that interval.

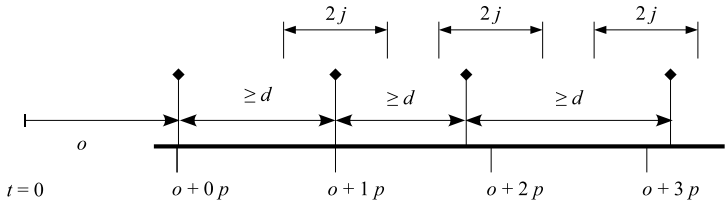


Figure 12.3: Workload definition using the (p, j, d, o)-notation

12.3 Modular Performance Analysis

Modular Performance Analysis (MPA) was developed by Thiele et al at ETH Zürich (see [111, 29]). MPA belongs to the class of so-called deterministic queuing theories. These models can be solved analytically (without simulation). Systems are modeled as a set of hierarchical queuing networks, as shown in Figure 12.4 for the combination *ChangeVolume* and *ReceiveTMC*. The workload of the system is described by a pair of interval bound functions (α), the so-called lower and upper *arrival curves* α_l and α_u. Suitable arrival curves can be constructed for any (p, j, d, o)-tuple. These curves describe the respective bounds on the number of events that are to be handled by the system for any given interval size. The available resource capacity (β) is characterized by a pair of lower and upper *service curves* β_l and β_u. These curves describe the bounds on the available capacity of the resource for any given interval size. Each component in Figure 12.4 delivers a pair of output arrival curves describing the event rates after this processing step and a pair of output service curves which describe the remaining capacity.

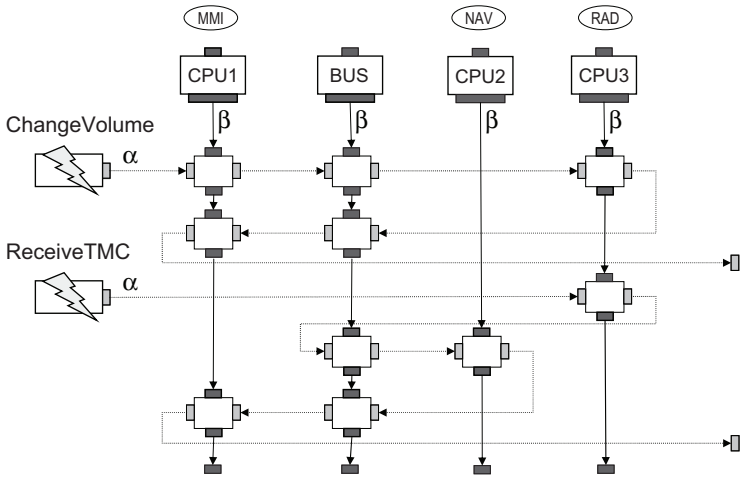


Figure 12.4: Example MPA queuing network for Architecture (a)

Analysis of the network provides us with answers to propagation delay and resource usage for each component individually as well as end-to-end. In addition, the backlog (the number of outstanding events which corresponds to maximum queue size needed) can be determined for each component. MPA provides *hard* bounds to all these properties, which makes it suitable to analyze hard real-time systems. But these bounds are *not* necessarily tight. Because the method works in the time interval domain, some information is lost in the transformation which may lead to pessimistic results. For example when a phase shift exists between two otherwise independent event streams. However, evaluation of an MPA network is very fast (typically a few seconds at most)

which supports the interactive nature of the design process. An open source implementation of MPA in Java for Matlab/Simulink is available from *http://www.mpa.ethz.ch*. A detailed treatment of MPA and this case study is provided in [121].

12.4 Symbolic Timing Analysis for Systems

Symbolic Timing Analysis for Systems (SymTA/S) was developed by Ernst and Richter and co-workers at the University of Braunschweig. Tool support is further developed at the SymtaVision company. It is a performance and timing analysis tool based on formal scheduling analysis techniques and symbolic simulation. It supports modeling of heterogeneous architectures, complex task dependencies, context aware analysis and combines optimization algorithms with sensitivity analysis for rapid design space exploration.

The tool provides a graphical user-interface to enter the model. Tasks can be (re-)assigned to resources by drag-and-drop. The environment is modeled by connecting event generators to the initial tasks. Properties of each entity in the model can be changed easily by means of pop-up menus, for example to modify the (p, j, d, o)-values of the event generators. Incompatible interface connections or requirements that are not met after analysis are made visible to the user graphically, by changing the color of the entity in the diagram that caused the error.

Evaluating a model is quick, typically in the order of a few seconds to a minute. The tool computes the local optimum per resource using classical formal scheduling analysis techniques like, for example, rate monotonic analysis. The values obtained for each resource are used to feed a symbolic simulation step where system-level values are derived. Using optimization strategies, this process is repeated automatically until some, user defined, property is reached. Like MPA, SymTA/S gives hard but not necessarily tight results, which is primarily caused by the abstractions introduced in the model. SymTA/S is available, as a commercial product, from *http://www.symtavision.com*.

12.5 Timed Automata and UPPAAL

The timed automata language [4] is a general purpose notation used to describe timed systems. An automaton consists of *locations* and *transitions*. Time can be modeled by introducing *clocks* as state variables. Clock *invariants* can be added to a location. The transitions define how those locations can be reached starting from some initial location. Transitions can be labeled with *guards*, for example to specify for which clock or state value(s) a transition is enabled. This technique is useful for our purpose mainly because of the expressiveness offered.

The UPPAAL model checker [11] is used to analyze the timed automata model. UPPAAL was developed by Yi and Larsen et al at Uppsala and Aalborg University respectively. UPPAAL provides a graphical user-interface to compose and edit timed automata models. A simulator is provided to animate the specification. The model

checker performs a symbolic exhaustive search over the state space in order to verify some user-defined property. If the property does not hold, a counter example is automatically generated which can be visualized and animated.

Hendriks showed in [57] that it is indeed possible to model our case study using timed automata. The principle idea of the model is that system resources are either *idle* or performing some task, i.e. computation or transferring data. Resource activity is modeled as a location. Transitions are defined from the *idle* (initial) location to each of the activity locations and vice-versa. The outgoing transitions are guarded by a counter which represents the number of outstanding requests for a particular activity. The counters are used to model the interaction between the different resources. When such a transition is taken, one is allowed to stay in the target location for the amount of time that corresponds to the user-defined maximum execution time of that task. When this time is reached, a transition back to the *idle* state is taken. Pre-emption of tasks can also be modeled and template automata are defined to describe the environment of the system.

The system model is constructed by composing a network of timed automata from the resource and environment automata described above. UPPAAL is then asked to verify whether a certain response time is within the set of reachable states of the model. By using a binary search approach manually, the exact best and worst-case response times can be determined. Evaluation of the model is in the order of minutes if the state space is tractable; the values then found are hard *and* tight. Tractability however, is mainly determined by the amount of non-determinism in the model. For example, when two event streams have an average period which is orders of magnitude apart, the state space explodes even though the model of the system is very small and simple. In this case, the model checker will not be able to find an answer in an acceptable amount of time. UPPAAL is available for free download from *http://www.uppaal.com*.

12.6 Parallel Object-Oriented Specification Language

The Parallel Object-Oriented Specification Language (POOSL) is a general purpose specification language which lies at the core of the Software/Hardware Engineering (SHE) system-level design method. POOSL was developed by Voeten and Van der Putten et al at the Technical University Eindhoven [95]. The language contains a set of powerful primitives to formally describe concurrency, distribution, synchronous communication, timing and functional features of a system into a single, high-level, executable model. Its formal semantics is based on timed probabilistic labeled transition systems. The SHE method is accompanied by two tools, SHESim and Rotalumis. SHESim is a graphical environment intended for incremental specification, modification and validation of POOSL models. Rotalumis is a high-speed execution engine, enabling fast evaluation of system properties by means of simulation. A more elaborate discussion on POOSL can be found in Chapter 13.

De Hoon constructed a model of our case study in [40] using this technique. Evaluation of this model is in the order of minutes to hours, depending on the property to

analyze. Despite the fact that the simulator conforms to the formal semantics of the language, no guarantee can be given that the model is completely covered during simulation. Exact best- and worst case values are not necessarily found during analysis, i.e. the bounds found by the simulator are *not* hard. Exhaustive analysis techniques are available but have not yet been implemented into tools. These exhaustive analysis techniques are subject to the state-space explosion problem, just like UPPAAL. However, POOSL is able to describe and analyze the nominal (average) behavior of the system and complex system - environment interactions, for example involving timing dependencies between input stimuli. POOSL is well-suited for analysis of soft real-time systems. The tools are available for free download from *http://www.es.ele.tue.nl/poosl/*.

12.7 Results and discussion

The case study was modeled using several techniques but the question is: How do the answers found during analysis relate? Consider for example the system-level response time for each of the applications in the case study. Based on the properties of the techniques themselves, we would expect to find results as depicted in Figure 12.5. MPA and SymTA/S provide hard but not necessarily tight bounds for these values. The approximations inherent to these methods may yield conservative results. Simulation based techniques, such as POOSL, also do not provide tight bounds because the model is not guaranteed to be fully covered, which may lead to results that are too optimistic. Timed automata can find hard and exact bounds within a user-defined accuracy, but only if the state space remains tractable. Tractability is, however, not a given fact. Although it is fairly easy to inspect timing aspects using timed automata, it is very hard to analyze the resource usage per resource. The only guarantee we have is that none of the resources is over-allocated. The other three methods in comparison do provide more detailed resource usage information and almost for free.

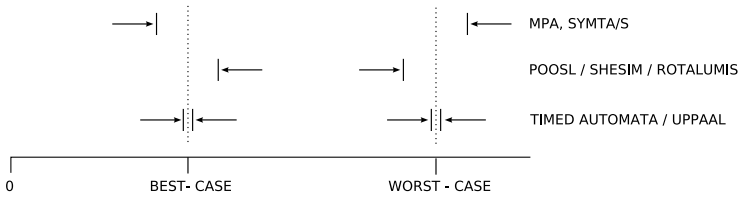


Figure 12.5: A general comparison of results found

Modeling comes at a price, there is a clear trade-off between abstraction and accuracy. How much effort do I need to invest in order to get a result on time and within a certain error margin? Can we determine this error margin at all a priori? Both MPA and SymTA/S are methods that are clearly tailored to support early life-cycle decision making. Models are easy to construct and evaluate. Suitable levels of automation are available for design exploration and sensitivity analysis. As we learned from addi-

tional experiments at Océ, not reported in detail here, their weakness is support for time dependent input stimuli. If these are dominant in your system, evaluation of models using these techniques typically leads to results that have little value when making design decisions.

Building timed automata and POOSL models by comparison involves significantly more work than MPA and SymTA, although modeling templates were developed to overcome this problem in part (see [44, 57]). Furthermore, analysis takes more time and is in general not guaranteed to lead to results. In the case of timed automata, expert knowledge may be required to modify the model such that tractability is achieved. However, the models built with these techniques can be described in much greater detail if needed, for example to deal with time dependent input stimuli, but obviously at the cost of model analysis efficiency. POOSL can provide feedback on the nominal (average) system behavior while the other three techniques can only investigate the performance bounds.

Table 12.1 provides an overview of the analysis results for the worst-case response time of all applications deployed on architecture (a) in Figure 12.2. The abbreviations *K2A* and *A2V* mentioned in the table refer to the system-level performance requirements of the *ChangeVolume* application shown in Figure 12.1. The models were evaluated against pure periodic environment stimuli with an unknown offset between the two event streams. This situation enables a fair comparison of the results because it can be suitably analyzed by all techniques.

Tool / *Requirement*	*Uppaal*	*POOSL*	*SymTA/S*	*MPA*
TMC + Volume	381.63	366.94	382.09	390.09
TMC + Address	239.08	234.26	253.30	265.85
K2A (Volume + TMC)	27.72	27.71	27.72	28.16
A2V (Volume + TMC)	41.80	41.78	41.80	42.24
Address + TMC	79.08	78.90	79.08	84.07

Table 12.1: Worst-case response time results (in msec)

The parameters in the case study were chosen such that the *ChangeVolume* application always has the highest priority on all resources. Because fixed priority pre-emptive scheduling is used on all resources, this application gets access to the resource as soon as it is required. This is why the last three rows of the table contains almost identical results for each method. In fact, we can calculate it by hand and we find the same values, which demonstrates the validity of the results found by the tools. When we compare the first two columns, we see that the POOSL results are indeed slightly more optimistic than the values found by UPPAAL. This is due to the fact that there are infinitely many possible values for the offset. In this particular case, UPPAAL was able to handle this property symbolically. It is also clear that both SymTA/S (third column) and MPA (fourth column) are slightly more conservative than UPPAAL, as expected.

Comparing the analysis results showed us that each technique introduces hidden

assumptions and approximations of its own. To our surprise, the initial results did *not* conform to the expectation illustrated in Figure 12.5. A discussion was started on the meaning of the results, to gain more insight. Apart from a better problem understanding, this also included improving the case study specification, discovering subtle modeling errors and even bugs and limitations in the (prototype) analysis tools. Table 12.1 is the result of *several iterations* due to this debate.

12.8 Conclusion

We have investigated four state-of-the-art techniques for performance analysis. We showed how they relate by means of an experiment. More results are available, for example using QOSA [17] and VDM++ [117]. We also performed measurements on a real system which are not yet considered here. However, our aim was not to be exhaustive in our survey. Many other existing techniques have not been considered and only a single case study was used for comparison. Neither did we attempt to determine what the "best" method is, since this is very context dependent. Many qualities influence failure or success. However, in conclusion, we do argue that in-depth knowledge of the application domain, the method used and awareness of the limitations of the tools are *equally important* critical success factors. This seems obvious, but in practice it is hardly ever the case that all three aspects are covered to the same extent. The small experiment has clearly demonstrated that it does pay off to use more than one method. Weaknesses in the models will be exposed by comparing the models and the analysis results. The models, the tools and the analysis results should not be taken for granted.

Acknowledgments We wish to thank Ernesto Wandeler, Simon Perathoner, Kai Richter, Martijn Hendriks, Menno de Hoon, Egor Bondarev, Peter Gorm Larsen and Erik Oosterom for their contribution to this survey.

Chapter 13

Model-driven design of real-time systems

Authors: O. Florescu, J.P.M. Voeten and H. Corporaal

The main purpose of engineering models is to help engineers understand the interesting aspects of a future system, before getting to the expense and trouble of actually building it. Traditional forms of engineering (e.g. mechanical engineering, electrical engineering) have a well-developed modeling methodology and their use of models is generally recognized as a useful and effective technique. However, software engineering, and particularly real-time embedded software, is still an emerging discipline. It is used for increasingly complex systems and its modeling techniques are neither mature nor reliable yet. Nevertheless, software models have a unique and remarkable advantage: they could be used to automatically generate executable programs for particular platforms. Starting with a simplified and highly abstract model, which must cover for both timeliness and functionality (e.g. architecture structure, concurrency, communication), refinements should be carried on until a complete specification is obtained, including all the details necessary in the final product, and from which adequate computer tools can generate an implementation. Mathematical techniques must assist each step from this trajectory from software model to its implementation to guarantee the correctness of the final system and its compliance to the requirements.

To support the model-driven development of software systems, the Unified Modeling Language (UML) [87] has been adopted as a standard facility for constructing models of object-oriented software. UML has proven to be suitable for modeling functional aspects of a system and there are defined extensions to it to provide a standardized way of denoting timing aspects for real-time systems [88]. Nevertheless, application of mathematical analysis techniques remains complicated due to the difficulty of relating formal techniques to UML diagrams [61]. Moreover, due to the lack of a standard formal semantics, a clear relation between the properties of a UML model and

of its generated implementation cannot be established. Hence, properties of the system implementation cannot be predicted from the model.

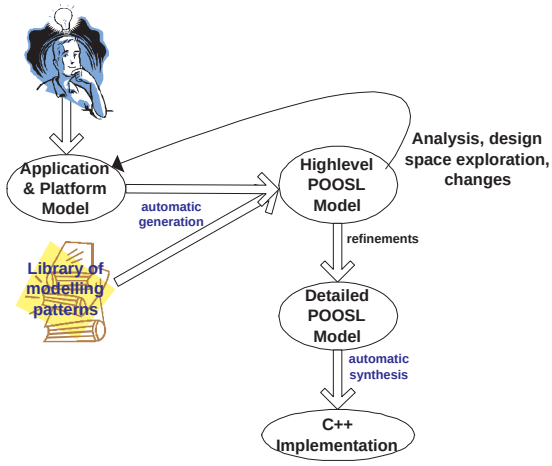


Figure 13.1: Model-driven design approach

The Parallel Object-Oriented Specification Language (POOSL) [52], which lies at the core of a system-level design method called Software/Hardware Engineering (SHE), is a mathematically defined modeling language that contains a set of powerful primitives to formally describe timing, concurrency, probabilistic behavior, (synchronous) communication and other functional features of a system into a single executable model. Its semantics is based on timed labeled transition systems, which guarantees a unique and unambiguous interpretation of POOSL models. Due to its formal semantics, POOSL is suitable for specification and verification of correctness and analytical computation of performance for real-time systems. Based on the concepts of this language, we have developed a library of modeling patterns, from which we can automatically generate the POOSL model of a system, even in early stages of the design trajectory, as shown in Figure 13.1. Such a model is amenable to analysis, design space exploration and trade-offs, based on which decisions for changes and/or refinements can be made. When all the necessary details are in the model, automatic generation of C++ implementation that preserves the properties of the model is realized.

In this chapter, we will show that a model-driven design approach, based on POOSL and its related techniques, is able to yield, in a fast way, correct implementations of real-time systems that satisfy their requirements. The following subsections present the steps of this approach. Section 13.1 describes the modeling and analysis phases, whereas Section 13.2 discusses the automatic synthesis of C++ implementation. An educational case study is presented in Section 13.3, and conclusions are drawn in Section 13.4.

13.1 Models and analysis for real-time systems

Complex real-time embedded systems are usually comprised of a combination of hardware and software components that are supposed to synchronize and coordinate different processes and activities. From early stages of the design, many decisions must be made to guarantee that the realization of such a complex machine meets all the functional and non-functional (timing) requirements. To properly deal with such issues, system models are built and analyzed to predict the properties of the final realization and to find the most suitable hardware platform that enables the system to meet the requirements.

13.1.1 Real-time properties

One of the purposes of real-time systems analysis is the verification of its *properties*. These properties are related to (observable) actions of interest that must occur at certain moments in time. Such properties can be formalized using temporal logics, like for example Metric Temporal Logic (MTL) [71].

Due to its amenability to mathematical analysis techniques, as a modeling language, POOSL is a good candidate for building real-time systems models. The timing semantics of a POOSL model is based on a two-phase execution model [85]: the state of the system changes either by asynchronously executing simultaneous atomic actions based on the interleaving semantics, without passage of time (phase 1), or by letting time pass synchronously for all the components of the system when no action can be performed (phase 2). As soon as an action becomes available, the first phase is resumed. Such a model assumes an abstract notion of time, called *model* time.

Based on such a model, the behavior of the system is considered to be the set of all the possible sequences of actions together with the time stamps at which they occur. These sequences are called timed action sequences. Their analysis reveals if there is any deadlock and if the system exhibits certain real-time properties. The behavior of a system satisfies a real-time (observable) property expressed as a temporal logic formula if and only if all possible timed action sequences in the behavior of the system satisfy that property. In other words, for the type of properties that we are considering in this work, the corresponding observable action of interest occurs in all sequences and at the same time stamp.

In order to establish a relation between timed action sequences in a model and in an implementation, a notion of observable distance between two different timed action sequences that have the same sequence of observable actions was introduced in [41]. This distance represents the largest deviation between the time stamps of corresponding observable actions. Two timed action sequences whose distance between them is equal to ϵ are called ϵ-close. When one of the sequences satisfies an observable real-time property P, namely that a certain observable action occurs in an interval $[t1, t2]$, the other timed action sequence satisfies a weakening of this property up to $2 \cdot \epsilon$. Thus, the observable action of interest occurs in the second timed action sequence in a larger

interval, $[t1-2\cdot\epsilon, t2+2\cdot\epsilon]$. This result was mathematically proved in [62] and, based on it, a property preserving code generation mechanism was conceived and implemented (see Section 13.2).

13.1.2 Design space exploration

Another purpose of system models analysis is to find the most suitable hardware platform that enables the satisfaction of all the requirements of the system, like real-time properties, cost, et cetera. Often, some of these requirements are in contradiction with each other, like fastest response, which typically implies an expensive platform, and low cost. Exploration of the design space and design trade-offs must be made in order to find a good balance.

One of the approaches for performing systematic design space exploration is the Y-chart scheme, introduced in [69]. This scheme makes a distinction between applications (the required functional behavior) and platforms (the infrastructure used to perform this functional behavior). Although we are concerned only with the realization of the software part of a real-time system, the hardware part must also be taken into account in the analysis in order to predict the behavior of the system as a whole and the impact each part may have on the others. Moreover, as real-time systems are typically reactive systems, meaning that there is a continuous interaction with the outside world, in [43] we added the model of the environment to the Y-chart scheme, as depicted in Figure 13.2. The design space can be explored by evaluating different mappings of the application onto platforms, under certain behavior of the environment.

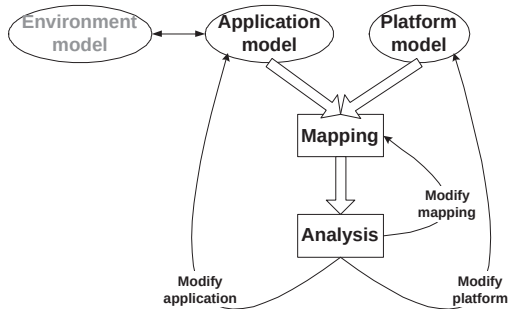


Figure 13.2: Y-chart scheme for design space exploration

As the components of real-time embedded systems usually have common characteristics, like tasks, computation/communication resources, we have developed a library of modeling patterns (also called templates) that can be used to automatically generate and easily modify models of the same or of a similar system. These patterns were conceived based on design experience from several case studies (see [44, 40, 39], and Chapter 8) and they are presented in Table 13.1. This table shows the Y-chart component to which each of these patterns belongs, the name of the pattern and its parameters. A brief explanation of the patterns is given below, whereas more details can be found in [42].

Y-chart Part	Pattern Name	Parameter Names	
Application Model	PeriodicTask	period (T) deadline (D) BCload WCload	latency (l) iterations loadDistribution
	AperiodicTask	deadline (D) BCload WCload	latency (l) loadDistribution
Platform Model	Resource	initial latency	throughput
	Scheduling	scheduling policy	
Environment Model	Environment	arrival stream upper bound (u)	lower bound (l)

Table 13.1: Modeling patterns

The application model of a system is described as a collection of real-time tasks, each characterized by the deadline, the load (which represents a certain distribution between a best-case and a worst-case value of the number of instructions that the task needs to execute at each activation), and the latency of task activation. Based on the type of activation request, tasks can be periodic (time-driven), being activated at regular intervals equal to the task period T which becomes a parameter of the pattern, or aperiodic (event-driven), waiting for the occurrence of a certain event.

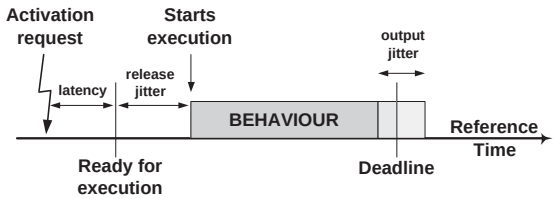


Figure 13.3: Real-time task parameters

In the task model, we are able to take into account three types of uncertainties, as shown in Figure 13.3, which are explained in [42]. The classical real-time scheduling theory [27] can compute the release jitter and, to some extent, the output jitter, but without taking into account neither possible variations nor dependencies on the input data. Moreover, the activation latency, which is caused, for example, by the delays between the occurrence of an event in the environment and the trigger of a sensor, is ignored. In our modeling approach, we are able to take these aspects into account and, hence, the designer is able to check if, under different circumstances, the behavior of the system still meets the critical deadlines for the control of the physical components that 'run' relative to the reference time.

Another part of the Y-chart scheme is the platform model which consists of (computation and/or communication) resources that can uniformly be characterized by an

initial latency and a throughput, and the scheduling policies that handle the concurrent requests. According to some desired mapping, the mapping stage of the Y-chart is realized by connecting tasks to resources using POOSL channels. The environment, which can be characterized by an event stream with a certain distribution of arrival between an upper and a lower bound, triggers the behavior of the tasks and expects a reaction from the system at certain moments in time.

By specifying the necessary modeling patterns, from the ones presented in Table 13.1, together with their parameters, the complete model of a system is automatically generated, as shown in Figure 13.1, and can be validated. For each configuration specified and generated, during the execution of the model, the resources schedulers may report if there are tasks that missed their deadlines. Furthermore, based on the POOSL formal semantics, it can be detected if there is any deadlock in the system. If all the deadlines are met and there is no deadlock, then the corresponding platform is a good candidate that meets the system requirements. In case of soft real-time systems, where a certain deadline miss ratio is allowed, the analysis of the model can handle and record tasks with multiple active instantiations that have missed their deadlines. The percentage of deadlines missed can be monitored and checked against the requirements if, according to this criterion, the underlying platform is suitable. The analysis of the model can also provide the release jitter, the output jitter and the number of instances each task has active at the same time.

Moreover, such models can be used to estimate the time-deviation the implementation will exhibit from the model when the code is generated on the platform. As actions in the semantics of the model are timeless, no matter how fast the platform is, a time-deviation will appear between model and implementation. If the designer considers the time-deviation to be too large and hence the properties would be weakened too much in the implementation, this time-deviation needs to be accounted for in the model, as shown in [43].

13.2 Synthesis for real-time

As mentioned in the previous section, a real-time system can be viewed as a set of timed action sequences. Moreover, both the model and the realization of a system are viewed as such sets. Hence, to obtain an implementation of a system which preserves the properties analyzed in its model, two things must be achieved: (i) to generate only traces which are included in the set of timed action sequences of the model; (ii) to make the corresponding traces in the model and in the implementation to be ϵ-close.

In this section, we briefly present a model synthesis approach based on the concepts of POOSL language that was implemented in a tool called Rotalumis-RT. The data part of a model, which refers to the information that is generated and exchanged by the active components of the system, is directly translated into C++ code. To obtain a trace that has the same sequence of observable actions as in the model, for the process part, representing the active components, process execution trees (PETs) [15] were adopted. The state of each process is represented by a tree structure, where each leaf is a

statement and internal nodes represent compositions of their children. The correctness of PETs with respect to the semantics of the POOSL language was formally proved in [51]. Details about PETs implementation and behavior can be found in [15].

Using a proper design time annotation of the model to distinguish between observable and unobservable actions, during the evolution of the system, PETs can send *observable action requests*, *unobservable action requests* and/or *delay requests* to a PET scheduler. The PET scheduler, whose behavior is described by the algorithm in Figure 13.4, asynchronously grants all eligible atomic observable actions. When no observable action is eligible, one unobservable action request is granted, and then the observable action requests list is checked again. When no action request of any kind is available, time passes synchronously for all PETs until some action becomes eligible again. As a result, the generated implementation exhibits exactly the same behavior as the model, if interpreted in model time domain. On the other hand, since the progress of model time is monotonically increasing, which is consistent with the progress of physical time, the action order observed in model time domain is consistent with that in physical time. To obtain the same (or similar) quantitative timing behavior in physical time as in model time, the PET scheduler tries to synchronize model time with physical time during the running of the implementation. This ensures that the execution of the implementation is always as close as possible, under the given circumstances, to a trace in the model with respect to the observable distance between timed action sequences.

```
PETSCHEDULER()
 list observableActions;
 list unobservableActions;
 list delays;
 while true do
  while observableActions.notEmpty() do
    observableActions.getAsynchronously()–>grant();
  if unobservableActions.nonEmpty() then
    unobservableActions.getAsynchronously()–>grant();
    continue;
  else
  if delays.nonEmpty() then
    modelTime = modelTime + delays.getFirst()–>amountOfTime();
    /* synchronisation between model and physical time */
    wait_until physicalTime == modelTime;
    continue;
  else
    DEADLOCK();
    return.
```

Figure 13.4: The PET scheduler

13.3 Educational case study

To illustrate the steps of the model-driven design approach described in the previous sections, we considered the control of a motion system made of two devices running in parallel as a case study (see Figure 13.5). Such a system is representative, for example,

for the control of a part of a printer where several motors must be controlled concurrently. The control algorithms that ensure stability of the system were designed by control engineers. The goal of this experiment was the application of the SHE model-driven design method for the development of the real-time system that is able to control correctly these devices.

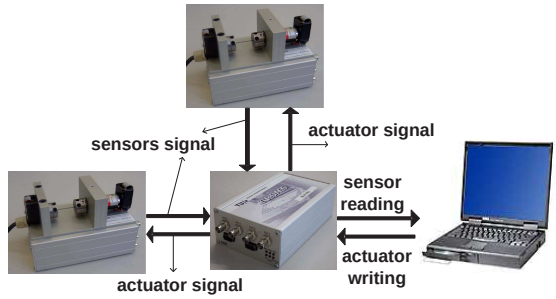


Figure 13.5: The setup of the case study

An analysis model of the motion system is shown in Figure 13.6. The software part of the system is made of two parallel aperiodic event-driven tasks, $MotorController_1$ and $MotorController_2$, which are mapped onto a single $Processor$ resource, whereas the environment consists of $Motor_1$ and $Motor_2$. The code shown in Figure 13.6 is the POOSL model for each $MotorController$. The method $controlAlgorithm$ models the actual control algorithm for a motor that has been designed by control engineers. The deadline D was set to $1ms$ for the first motor, and to $2ms$ for the second motor.

To ensure the stability of the control of the two motors system, two required real-time properties must be satisfied by the application part of the system. For the first motor, the message $actuatorOutput$ must *always* be sent between 0.9 and $1.1ms$ after the message $sensorInput$ is received. For the second motor, the property that needs to be satisfies is that the message $actuatorOutput$ must *always* be sent between 1.9 and $2.1ms$ after the message $sensorInput$ is received. As the model is very simple, assuming that the execution time of the control algorithm of each device on the given processor is $0.4ms$, we could manually check that for the first motor the message to the actuator is sent after $1ms$, whereas for the second motor, after $2ms$, which means that the model satisfies the required real-time properties. Moreover, assuming that a communication operation takes $0.01ms$, we estimated a time-deviation of $0.04ms$ between model and implementation, because there are at most four communication operations that occur at the same model time.

To enable automatic generation of an implementation of the application part, a synthesis model was developed. In this model, the environment and the platform parts were removed and all the communication with the environment was replaced with a synthesizable interface. The model presented in Figure 13.7 shows how the communication with the environment model was replaced with calls to the $readSensor$ and $writeActuator$ methods from a data class called DAS - Data Acquisition System. This data class provides only virtual methods for the synthesis model. Its actual implemen-

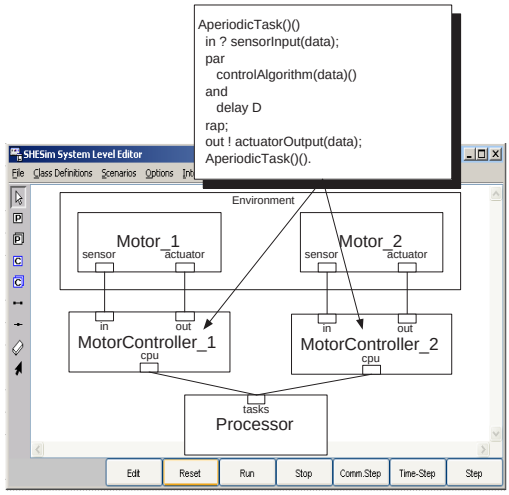


Figure 13.6: Model of the system case study

tation for the communication with the real motors via a special device that can be seen in Figure 13.5 is provided by Rotalumis-RT in C++ code. The implementation of the interface should not be blocking because its timing behavior can affect the deviation between model and implementation.

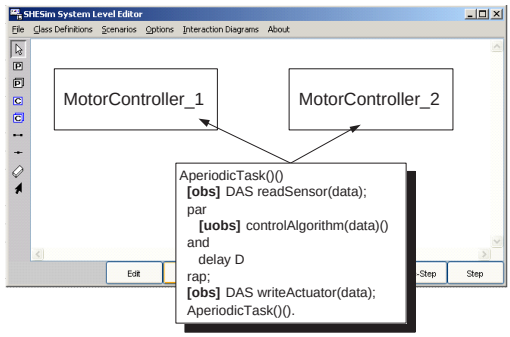


Figure 13.7: Synthesis model of the system

We have labeled each action in the model such that Rotalumis-RT could identify which is considered observable or unobservable. Actions like reading from the sensor and writing to the actuator are observable activities of the system, whereas the control algorithm computation is unobservable from outside the system. During several hours of continuous behavior, the maximum obtained observable distance between model and the generated implementation was $0.042ms$. This value is larger than the

estimation from the model because of the overhead of the operating system that was running on the computer on which the synthesis of the model was realized. Thus the properties satisfied by the implementation are that the communication with the first motor takes place in an interval $[0.916, 1.084]ms$ and for the second motor within $[1.916, 2.084]ms$, which fulfills the requirements. Hence, the real-time system presented as case study could be developed in a model-driven fashion that ensured the satisfaction of the system requirements.

13.4 Conclusions

In this chapter, we have presented a model-driven design approach for real-time systems based on a formally defined modeling language named POOSL. For the analysis stage of this approach we rely on the mathematical definition of the language. We have developed a library of modeling patterns that enable automatic construction of the model. Moreover, we have presented a mechanism for automatic code generation that enables the preservation of the properties analyzed in the model. An educational case study shows the application of each of the steps of this model-driven design approach and the results that we have obtained. As future work, we aim at improving the synthesis step such that it can be applied to more complex systems that may even incorporate time-intensive computations.

Chapter 14

Time-varying delays in control

Authors: M.B.G. Cloosterman, N. van de Wouw, W.P.M.H. Heemels and H. Nijmeijer

14.1 Introduction

During high-tech systems design, the different couplings between the domains of mechanics, electronics and software have to be considered. In these couplings conflicts exist in the domain-specific properties and requirements. Similar to Chapter 16, this chapter deals with the coupling between control engineering and real-time software design that is apparent in many high-tech systems. Here, the focus is on the latency and jitter, which is inevitable in the software implementation of a controller and affect the performance of the controlled system (plant), e.g. a motor in the paper path of a printer. From a control point of view, time-delay, consisting of the combination of both the latency and jitter, which includes computation times, communication delays and probably a reaction time of the sensors or actuators, is an undesired phenomenon that should be kept as small as possible. In control engineering, it is well known [46] that these time-delays can degrade the performance of the controlled system and can even cause instability of this system. In practice, in many motion control applications the time-delay is assumed to be negligible compared to the chosen sample-frequency or it is assumed to be constant. From a software point of view, latency and jitter can not be avoided, and even worse, can not be predicted accurately [27, 42]. It is known that latency and jitter are affected by various aspects that are related to the software and its hardware, such as caches, pipelines, the characteristics of the software architecture and the chosen communication network, e.g. a CAN-bus or ethernet [86]. In general, the combination of latency and jitter results in time-variations in the moment of actuation of the controlled system, when compared to the sample moment of the measurement being used in the feedback. This contradicts the general assumption on zero or constant

time-delays that is often made in control design. Schematically, this conflict between the disciplines is depicted in Figure 14.1(a).

In this chapter, we describe a first step to incorporate effects from software in the control design and vice versa by considering time-varying delays, instead of a design based on the general assumption that the time-delay is constant or even zero. Schematically, this connection is depicted in Figure 14.1(b). Now, in the coupling between control and software the demands on the maximum time-delay, for which a certain performance can be guaranteed, can be compared to the achievable latency and jitter in the software implementation. As depicted in Figure 14.1(b), the opposite direction is possible as well. Such a viewpoint allows us to make explicit the consequences of the design choices in one domain on the performance or requirements in the other domain, thereby allowing for a more integrated design trade-off process.

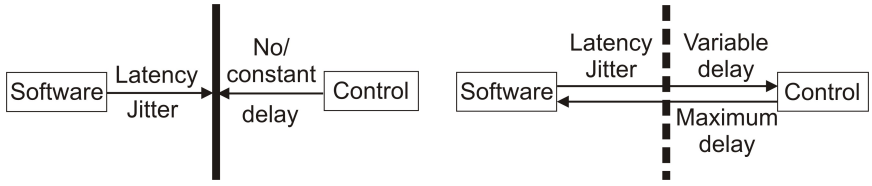


Figure 14.1: Schematic view of the coupling between real-time software and control engineering: (a) traditional viewpoint, (b) our viewpoint with mutual consideration of requirements.

In the literature, different examples are available where the time delays are taken into account during the controller design. A general approach is described in the field of Networked Control Systems (NCSs) [58], [127], [131]. In NCSs, the controller is coupled to the controlled system (physical plant) with sensors and actuators over a real-time network, as is depicted in Figure 14.2 [131]. Additionally, the information flows are given by dotted lines. Advantages of an NCS are that by using distributed elements, flexible architectures are obtained. In a copier/printer example, as used in the Boderc project, this might result in the use of one processor that computes the control actions for the different motors in the paper path, instead of using dedicated CPUs for each motor separately. A disadvantage is that all control-related data, i.e. measurement data and actuator data from different plants, and other data, such as software error notifications are sent over the same communication network. The transmission of these data causes time-delays, due to the waiting time until the network is empty and due to the transmission time of the network. Even worse, loss of data occurs in practice, because data packets may never arrive at the controller or actuator. Despite of these disadvantages NCSs have been used in different areas [112, 58], such as mobile sensor networks, remote surgery, automated highway systems, unmanned aerial vehicles,

robotic manipulators and teleoperation.

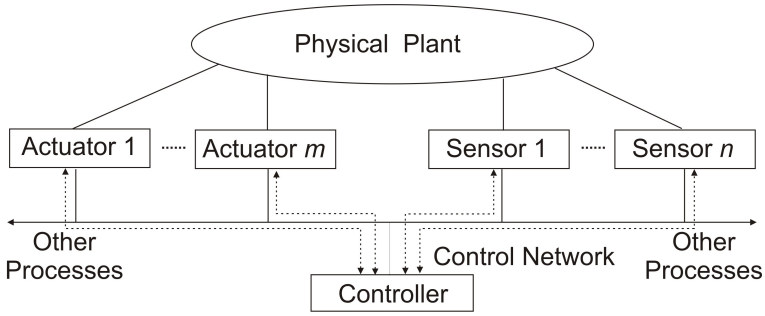


Figure 14.2: A typical NCS setup and information flows [131].

From literature, some results are known that describe the decrease of the performance for increasing delays. In [12], a simulation example is used to show that the use of a network in the control loop, changes the behavior of the controlled system. In their example, the influence of some network parameters on the step response (a specific transient performance measure [46]) is determined. The variable parameters are the network bit-rate (speed), the presence of disturbance traffic and its relative priority. This results in a different time-delay that has to be taken into account during the controller design. Their results are given in Figure 14.3, which shows that too small network speeds (cyan line, 10 kBps) or low priorities (blue line, p10) on the control signal result in an unacceptable decrease of the performance. Therefore, the latency and jitter need to be considered during controller design to a priori guarantee performance in the face of such delays. Other examples, where the influence of the time-delay on the system performance is investigated, are presented in e.g. [28, 75, 86, 112].

In the previous example, the decrease of the performance is obvious, but still the system is stable. A more dramatic result can be obtained if the variation in the time-delay destabilizes the NCS. Examples showing this effect are rare. In Section 14.3, we show that an NCS may become unstable for time-varying delays, varying within a bounded set; even when the NCS with any constant delay taken from this set is asymptotically stable. A similar example was also shown in [124].

14.1.1 Problem description

In the remaining of this chapter, we focus on the effects of time-varying delays on the stability of a control system. To avoid the occurrence of the destabilizing effect of time-varying delays in practice, two methods to determine the robust stability of an NCS are described. Here, robustness refers to robustness with respect to uncertain time-varying delays taken from a bounded set. For the sake of simplicity, first methods

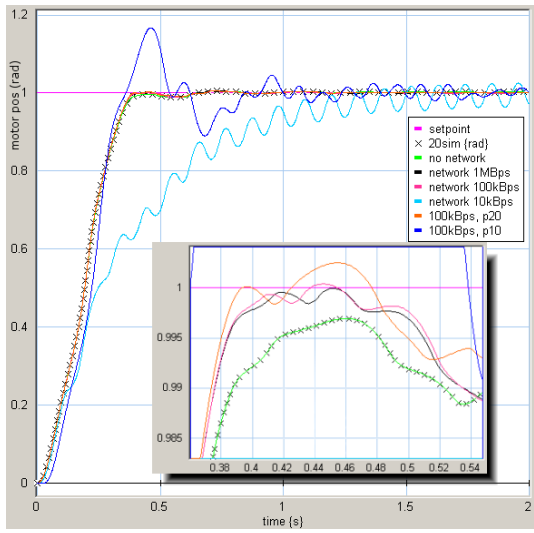


Figure 14.3: Several simulations of a setpoint control problem for a motor with different fieldbuses settings [12]. Inset: zoomed portion of the curve.

for time-varying delays upper-bounded by the sample-time of the control algorithm are obtained. The next step is to extend these methods for larger variations in the time-delay.

14.2 Basic NCS model

Different models for NCSs are available in literature. Roughly, they can be distinguished in discrete-time [54, 7, 124] and continuous-time descriptions [120, 130, 84]. A short description and comparison of the different models can be found in [31].

The model used in the remaining of this chapter is based on the description of an NCS, proposed in [7]. The NCS is depicted schematically in Figure 14.4. It consists of a continuous-time plant and a discrete-time controller that receives information from the plant only at the sampling instants $t_k = kh$ (with h the constant sample-time). In the model, also the computation time τ_k^c and networked induced delays, i.e. sensor-to-controller delays τ_k^{sc} and controller-to-actuator delays τ_k^{ca} are taken into account. Similar to [7], the sensor acts in a time-driven fashion and the controller and actuator (including the zero-order-hold (ZOH) in Figure 14.4) act in an event-driven fashion. 'Time-driven' refers to acting at the sampling instants and 'event-driven' refers to acting only if new information is available. Under these assumptions, in combination with a controller that is independent of the time-delays, and the assumption that vacant sampling does not occur ($\tau_k^{sc} < h$) all delays can be represented by a single delay $\tau_k := \tau_k^{sc} + \tau_k^c + \tau_k^{ca}$, which is taken into account in the discrete-time control sig-

nal u_k [86, 124]. The sampling instants t_k are determined by the time-driven sensor output. Moreover, we assume that the total time-delay τ_k is smaller than the constant sample-time h: $\tau_k < h$. The continuous-time model of the NCS can then be given by:

$$\begin{array}{rcl} \dot{x}(t) & = & Ax(t) + Bu^*(t) \\ u^*(t) & = & u_k, \quad \text{for } t \in [kh + \tau_k, (k+1)h + \tau_{k+1}), \end{array} \tag{14.1}$$

with A and B the continuous-time system and input matrices, respectively, $x(t) \in \mathbb{R}^n$ the state, $t \in \mathbb{R}$ the time, τ_k the delay at sampling moment k, and $u_k \in \mathbb{R}$ the delayed discrete-time input. For the sake of simplicity, we assume that we measure the entire state, i.e. $y_k = x_k$, at the sampling instants.

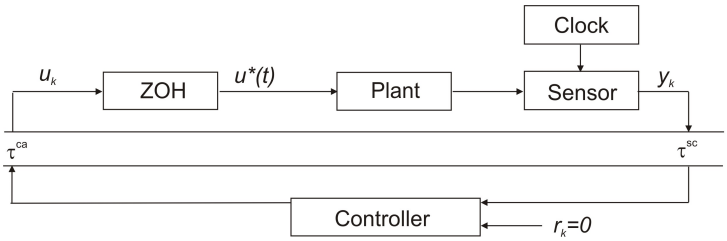


Figure 14.4: Schematic overview of the networked control system.

The discretization of (14.1) on the sampling instants $t_k = kh$ (the sampling moments) gives the discrete-time NCS model, which forms the basis of our analysis:

$$x_{k+1} = e^{Ah}x_k + \int_0^{h-\tau_k} e^{As}dsBu_k + \int_{h-\tau_k}^{h} e^{As}dsBu_{k-1}. \tag{14.2}$$

This equation is only valid at the sampling instants t_k, where the state is given by $x_k := x(t_k)$ and the related control action by u_k. In this work, we adopt a linear static state feedback law and the reference input of the feedback controller is assumed to be zero ($r_k = 0$ in Figure 14.4), which results in the control law $u_k = -Kx_k$. The closed-loop NCS model is then given by:

$$x_{k+1} = e^{Ah}x_k - \int_0^{h-\tau_k} e^{As}dsBKx_k - \int_{h-\tau_k}^{h} e^{As}dsBKx_{k-1}. \tag{14.3}$$

Now, by defining the state of the closed-loop NCS model by $\xi_k = \begin{pmatrix} x_k^T & x_{k-1}^T \end{pmatrix}^T$, we obtain the following state-space model, given $\tau_{max} \in [0, h]$:

$$\xi_{k+1} = \tilde{A}(\tau_k)\xi_k, \ \tau_k \in [0, \tau_{max}], \tag{14.4}$$

with $\tilde{A}(\tau_k) = \begin{pmatrix} e^{Ah} - \int_0^{h-\tau_k} e^{As}dsBK & -\int_{h-\tau_k}^{h} e^{As}dsBK \\ I & 0 \end{pmatrix}$, and $\xi_k \in \mathbb{R}^{2n}$. Note that in (14.4) arbitrary time-varying delays, upper-bounded by $\tau_{max} \leq h$, are accounted for.

14.3 A motivating example

Before analyzing stability of NCSs with time-varying delays, an example is given to show the effect of time-variation in the delay on the stability of the controlled system. For the sake of simplicity, in this section, we assume periodic variation of the time-delay.

The example is in the context of the document printing domain, see Chapter 1. In general, a paper path, consisting of pinches (rollers), driven by motors, is used to transport a paper through the printer. In this example, the motor controllers share the CPU-time of one processor, which is connected to the motors and sensors via a communication network resulting in unpredictable time-varying delays in the control loop. We zoom in on one single motor driving one pinch, as depicted in Figure 14.5. Still, the controller is connected to the motor via the network. In the motor-pinch model, the motor is assumed to behave ideally, the coupling between motor and pinch is assumed rigid and slip between the paper and pinch is neglected, which gives (see also [24]):

$$\ddot{x}_s = \frac{nr_P}{J_M + n^2 J_P} u, \tag{14.5}$$

with $J_M = 1.95 \cdot 10^{-5}$ kgm^2 the inertia of the motor, $J_P = 6.5 \cdot 10^{-5}$ kgm^2 the inertia of the pinch, $r_P = 14$ mm the radius of the pinch, $n = 0.2$ the transmission ratio between motor and pinch, x_s the sheet position and u the motor torque.

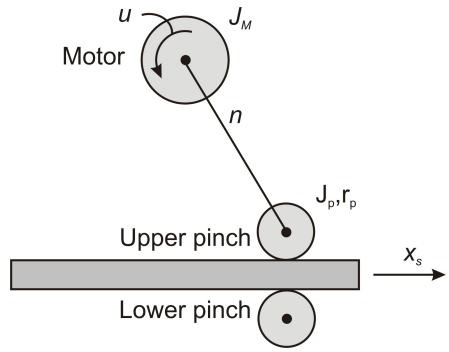


Figure 14.5: Schematic overview of the motor-pinch example

The continuous-time state-space representation of (14.5), where the delays are accounted for in the discrete-time input u_k is given by (14.1), with $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ \frac{nr_P}{J_M + n^2 J_P} \end{pmatrix}$ and $x(t) = \begin{pmatrix} x_s(t) \\ \dot{x}_s(t) \end{pmatrix}$. Adopting a feedback controller of the form $u_k = -Kx_k$, with $K = \begin{pmatrix} K_1 & K_2 \end{pmatrix}$, the integrals in $\tilde{A}(\tau_k)$ of (14.4) can be computed,

which yields:

$$\tilde{A}(\tau_k) = \begin{pmatrix} 1 - \frac{1}{2}\alpha^2 K_1 b & h - \frac{1}{2}\alpha^2 K_2 b & \tau_k \beta K_1 b & \tau_k \beta K_2 b \\ -\alpha K_1 b & 1 - \alpha K_2 b & \tau_k K_1 b & \tau_k K_2 b \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \tag{14.6}$$

with $b = \frac{nr_P}{J_M + n^2 J_P}$, $\alpha = h - \tau_k$, and $\beta = \frac{1}{2}\tau_k - h$.

If the delay τ_k is constant, and smaller than the sample-time h, the stability of system (14.4), with (14.6) can be determined by checking if the eigenvalues of $\tilde{A}(\tau_k)$ are inside the unit circle [7]. We consider this system with a sample-time $h = 1$ ms, and two possible constant delays: $\tau^a = 0.2$ ms and $\tau^b = 0.6$ ms. A linear feedback gain $K = \begin{pmatrix} 50 & 11.8 \end{pmatrix}$ results in a stable system for both constant delays τ^a and τ^b, as is illustrated by the upper plot of Figure 14.6. The eigenvalues of the matrix $\tilde{A}(\tau^a)$ are $\lambda_1 = 0.996$, $\lambda_{2,3} = -0.097 \pm 0.539i$, and $\lambda_4 = 0$. The eigenvalues of $\tilde{A}(\tau^b)$ are $\lambda_1 = 0.996$, $\lambda_{2,3} = 0.203 \pm 0.927i$, and $\lambda_4 = 0$. Moreover, the NCS of (14.4), (14.6) is stable for any constant delay τ chosen from the interval $[0, \tau^b]$. This is illustrated in Figure 14.7, where the region between the red lines represents the stabilizing controller gains K_2 for constant delays τ_{max} and $K_1 = 50$. Note that one red line is close to $K_2 = 0$.

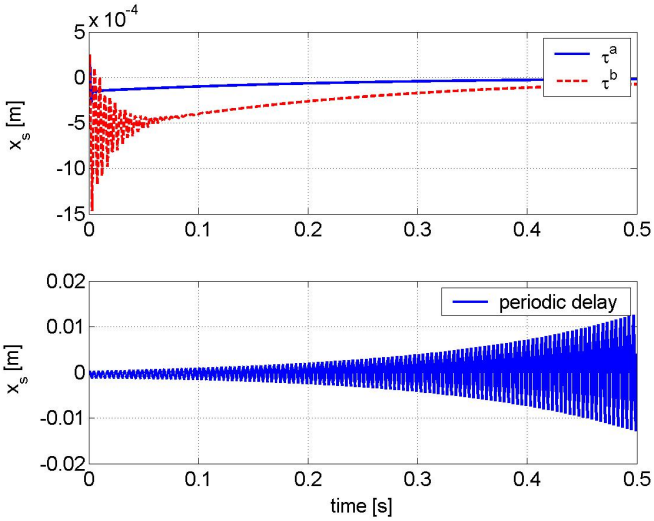


Figure 14.6: Time behavior of system (14.4), (14.6) for: (upper figure) constant $\tau^a = 0.2$ ms, $\tau^b = 0.6$ ms, and (lower figure) the alternating sequence τ^a, τ^b.

However, the system becomes unstable if the delays occur in an alternating sequence $(\tau^a, \tau^b, \tau^a, \tau^b, ...)$, as is shown in the lower plot in Figure 14.6. The instability

of this periodic system can be obtained from the eigenvalues of the matrix $\tilde{A}(\tau^b)\tilde{A}(\tau^a)$ [54], which are: $\lambda_1 = 0.992$, $\lambda_2 = -1.012$, $\lambda_3 = 0$, and $\lambda_4 = -0.267$.

In many practical situations, this periodic stability test is too limited. The use of the network results in variations in the time-delay, which are in general not periodic (see e.g. [86]).

14.4 Robust stability

As shown in the previous section, time-varying delays may result in instability. If, for a given controller, we can determine the maximum amount of time-delay (both latency and jitter) that is allowed, while still guaranteeing stability, this value can be used in the coupling between control and software designs.

A lot of research is performed on the stability of NCSs, recently. For the discrete-time models, stability results for constant and periodic time-delays are described in [54] and [7]. A thorough discussion of relevant stability results for time-varying delays can be found in [31]. General overviews of stability for NCSs are contained in [112, 127] and [131].

We propose a different approach here that is based on a direct convex embedding of the discrete-time NCS description in an uncertain system [32], which guarantees stability for the original system directly. This convex over-approximation of the discrete-time NCS model of (14.4) is using the concept of interval matrices. Based on this over-approximation the feasibility of the following Linear Matrix Inequalities (LMIs):

$$\begin{array}{lcl} P = P^T & > & 0 \\ \bar{A}^T P \bar{A} - P & < & 0, \ \forall \bar{A} \in \bar{\mathcal{A}}, \end{array} \tag{14.7}$$

with

$$\bar{\mathcal{A}} := \{\bar{A} \in R^{2n \times 2n} : \ \bar{a}_{ij} = q_{ij} \text{ or } \bar{a}_{ij} = r_{ij}, \ i,j = 1,2,...,2n\}, \tag{14.8}$$

with $\bar{a}_{ij}$ the $(i,j)^{th}$ element of $\bar{A}$, $q_{ij} = \min_{\tau \in [0,\tau_{max}]} \tilde{a}_{ij}(\tau)$ and $r_{ij} = \max_{\tau \in [0,\tau_{max}]} \tilde{a}_{ij}(\tau)$ the minimum and maximum value of the $(i,j)^{th}$ element $\tilde{a}_{ij}(\tau)$ of $\tilde{A}(\tau)$, respectively, guarantees the robust asymptotic stability of the networked control system for any time-varying delay $\tau_k \in [0, \tau_{max}]$, with $0 \leq \tau_{max} \leq h$. Here, the matrix $\tilde{A}(\tau)$ is equal to (14.4). A numerical disadvantage of this approach is the possibly large number of LMIs ($2^{\frac{m}{2}}$, with $m = 2^{2n}$, and n the dimension of the continuous-time system matrix A in (14.1)) that need to be checked for stability. We describe an improvement in [31], where a convex overapproximation, based on the Jordan canonical form of the continuous-time matrix A in (14.1), is used. Here, both the number of LMIs (2^n) and the conservatism of the method are reduced. For both methods, the region for which we can guarantee stability, based on the LMIs, is given in Figure 14.7. For small time-delays, both our methods do not seem overly conservative, compared to the stability region for constant delays (region between the red lines). For larger delays, it is obvious that the improvement based on the Jordan

canonical form (blue line) gives less conservative results than the ones based on (14.7) (magenta line). Additionally, the green line in Figure 14.7 represents the combination of the time-delays $\tau^a = 0.2$ and $\tau^b = 0.6$, as used in the motivating example of Section 14.3. Clearly, as expected, this combination is located in the region for which no stability could be proven based on both previously described methods.

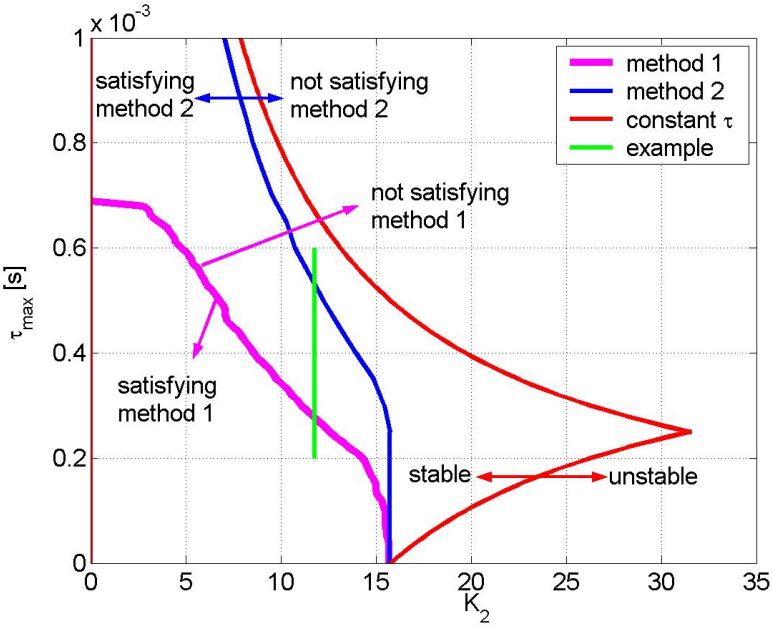


Figure 14.7: Stabilizing controller gains K_2 for time-varying delays in the interval $[0, \tau_{max}]$ based on the concept of interval matrices [32] (method 1) and based on the Jordan canonical form [31] (method 2), for constant delays equal to τ_{max} and the controller gain used in the example of Section 14.3 ($K_1 = 50$, $h = 1$ ms).

Of course, with our methods we only guarantee stability. Performance aspects, such as the real-time transient behavior (settling-time, overshoot) in Figure 14.3 and the effect of disturbances on the steady-state tracking error are not taken into account in the above described methods, but are part of future research. Moreover, we have no idea whether the maximum allowed time-delay τ_{max} (latency and jitter) is indeed achievable in the software implementation, but this value can be used in the discussion with real-time software engineering. The method proposed in Chapter 13 is useful to predict the amount of latency and jitter that is introduced in the implementation of the real-time software model on a certain architecture. Their software model is chosen such that it satisfies given real-time properties, such as the maximum time between the start and end of a control computation. Also some other methods that are presented in Chap-

ter 12 can be applied to estimate the latency and jitter that occur for a given software model. If the real-time software implementation is already available, another method is to obtain practical values by performing measurements on the implementation.

14.5 Large delays

Previously, only results for time-varying delays within the sample-interval are described. In many applications, delays larger than the sample-time or package loss occur. The model of Section 14.2, contains only delays that are allowed to take values in the interval $[0, h]$, with h the sample-time. In [7], an extension for larger, but constant delays is presented. A more general discrete-time model for time-varying delays larger than the sample-time, based on the previous model, is proposed in [128]. Here, additional parameters are introduced to describe whether or not a control signal, e.g. u_k or u_{k-1} in (14.2), is active in the current sampling interval. The number of active control signals in each interval is variable, due to the variation in the time-delays larger than one sampling interval. In [31], the previously described robust stability results are adapted such that they can be applied to a system that is faced with these larger delays. The increase of the number of variable parameters, due to the different number of control signals that can be active, results in an increase of the number of LMIs that have to be checked for robust stability. The number of these LMIs depends on the ratio between the maximum delay τ_{max} and the sample-time. For a more detailed description of this stability test and the obtained stability results the reader is referred to [31].

14.6 Conclusions and future work

In this chapter we discussed one particular aspect of the coupling between software and control designs: the latency and jitter that can not be avoided in the real-time software implementation but affect the performance and stability of the controlled system. Examples are presented that illustrate the deteriorating effect of such delays on the performance and the stability of the physical system. From a control point of view, these effects are analyzed and incorporated in the controller design. Therefore, the admissible latency and jitter, for which robust stability of the physical control system can be guaranteed, can be determined. These values need to be compared to the worst-case values of the latency and jitter, that can be derived from experiments or estimated based on the models explained in Chapter 12 and Chapter 13.

Based on the results for large delays, the influence of package loss can be analyzed, because the effect of package loss seems similar to the occurrence of time-delays that can be smaller and larger than the sample-time. This can be explained by the fact that the actuator will keep its most recent input until new data arrives, if a package is lost. Additionally, the results for package loss and time-varying delays, which may be larger than the sample-time, need to be combined to obtain a complete overview of the effect of the communication network on the stability of the control system.

Besides the effects on the stability, the effects on the performance are of interest. One particular aspect, is the real-time transient behavior, e.g. settling-time and overshoot, as presented in Figure 14.3. Another aspect is to analyze the influence of disturbances, such as measurement noise, on the steady-state tracking error between the reference signal and the output signal of the physical control system.

Chapter 15

Sheet feedback control in a printer paper path

Authors: B.H.M. Bukkems, J.J.T.H. de Best, M.J.G. van de Molengraft, W.P.M.H. Heemels and M. Steinbuch

15.1 Introduction

The design of a reliable sheet handling mechanism is a central issue in the development of today's cut sheet printer paper paths. An example of such a paper path is shown in Figure 15.1. Sheets enter this paper path at the Paper Input Module (PIM) and are transported to the Image Transfer Station (ITS) where the image is printed onto the sheet at high pressure and high temperature. After the print has been made, sheets can either re-enter the first part of the paper path for back side printing or they can go to the finisher (FIN). The transportation of sheets is done via pinches. A pinch is a set of rollers consisting of two parts: one part that is actuated by a motor and one part that is used to apply sufficient normal force to prevent the sheet from slipping. As can be seen from Figure 15.1, pinches can be driven either individually or grouped together in sections.

One of the objectives of the printer's sheet handling mechanism is to accurately deliver sheets to the ITS. Each sheet must synchronize with its corresponding image with respect to both the ITS entry time and the constant printing velocity to achieve a high printing quality. One way to realize the desired printing quality is using a high precision mechanical design. An alternative approach is to exploit the power of closed-loop sheet control. In this approach, the tolerances on the mechanical parts of the paper path are allowed to be larger and less effort and money have to be put in constructing a very stiff frame and drive train, since robustness against disturbances and parameter

uncertainties in the mechanical design is achieved by sheet feedback control. To realize a sheet feedback control system, the sheet position has to be known. This can, for example, be realized by adding position sensors, possibly in combination with model-based observer techniques.

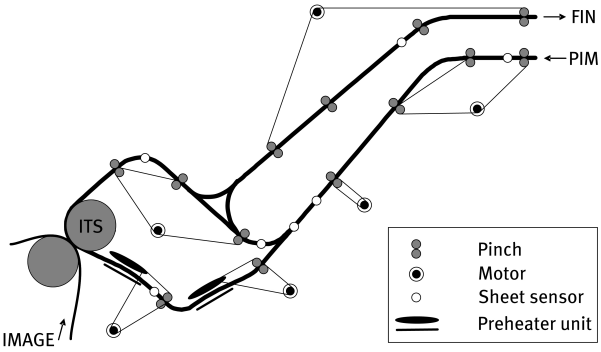


Figure 15.1: Schematic representation of a paper path.

As discussed in Chapter 6, besides the sheet reference profiles, the Happy Flow model also generates motor profiles that have to be tracked to realize these desired sheet profiles. Furthermore, the model takes into account requirements on motor characteristics and motor control algorithms. Hence, sheet schedules and information on motor dynamics and control are combined in one model. By introducing sheet feedback control, a decoupling can be made: the Happy Flow model will generate the sheet reference profiles, whereas the sheet control module will handle setpoint generation for the controlled motor dynamics, based on the actual sheet tracking error. The decoupling might lead to an increase in the design space for the sheet schedules, due to the decreasing number of details in the Happy Flow model.

Known results on sheet feedback control can be found in [73], [30], [97]. However, robustness against perturbations and disturbances is not explicitly taken into account in these control designs. The approach we propose is a model-based sheet feedback control design procedure in combination with a performance analysis to predict the effect of parameter perturbations. To synthesize controllers for the sheet tracking problem, we formulate the system in terms of its error dynamics. Experiments will show that a good tracking behavior has been obtained, also in case of parameter uncertainties present in the paper path.

The remainder of this chapter is organized as follows: in Section 15.2, the system under consideration will be discussed in more detail and the problem statement will be given. In Section 15.3, we will discuss the controller design method for the paper path system, together with the performance analysis in case of parameter perturbations. In Section 15.4, we will present the experimental setup that has been used to validate the proposed control design approach in practice. The validation experiments will be presented in Section 15.5, and conclusions and recommendations will come at the end.

15.2 Sheet feedback control problem

In this chapter, the focus will be on sheet feedback control design in a basic paper path, shown in Figure 15.2. By considering this basic version, the essence of the control problem becomes clear. As a result, the switching nature of the system, caused by the consecutive changing of the driving pinch, naturally arises in the control design and a structured design approach can be proposed. Since we consider the motion of sheets only when they are in the paper path, the PIM and FIN are not taken into account. The considered paper path consists of three pinches (*P1*, *P2*, and *P3*) only, each of which is driven by a separate motor (*M1*, *M2*, and *M3*, respectively). The locations of the three pinches in the paper path are represented by x_{P1}, x_{P2}, and x_{P3}, respectively. These locations are chosen such that the distance between two pinches is equal to the sheet length L_s, so the sheet can only be in one pinch at the same time. No slip is assumed to occur between the sheet and the pinches and the coupling between the pinches and motors is assumed to be infinitely stiff (i.e. kinematic). The mass of the sheet is assumed to be zero, which simplifies modeling of the sheet dynamics. The sheet position, defined as x_s, is assumed to be measured.

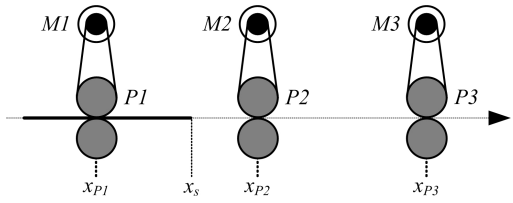


Figure 15.2: Schematic representation of the printer paper path.

We adopt a hierarchical, cascaded control structure for the sheet feedback control design. This control layout consists of low level motor control loops and a high level sheet control loop for tackling disturbances and uncertainties at the motor level and at the sheet level, respectively. The control goal we adopt for the basic paper path case study is the design of high level feedback controllers (HLCs) that track the desired sheet reference trajectory. Regarding this reference motion task, possible choices are absolute reference tracking control (ARTC) and inter-sheet spacing control (ISSC) [73], [30]. In this chapter, the first option is chosen, considering the eventual possible implementation in an industrial paper path as the one shown in Figure 15.1. Since this type of paper paths is often equipped with a registration unit where sheets are stopped for correction of their orientation and lateral position, implementing ISSC would lead to a standstill of all upstream sheets when a sheet is in the registration unit. Furthermore, a large inter-sheet spacing error between the sheet in the registration unit and its downstream neighbor will occur, leading to large control actions, i.e. large motor reference velocities. To avoid these undesired phenomena, ARTC is used and it is required that sheets are able to track a second-order sheet reference trajectory $x_{s,r}$, generated by the Happy Flow model.

The closed-loop linear motor dynamics in the Laplace domain can be represented by

$$\Omega_{Mi}(s) = T_i(s)\Omega_{Mi,r}(s), i \in \mathcal{I}, \tag{15.1}$$

with $T_i(s)$ the complementary sensitivity function of controlled motor i, which maps the input of the low level closed-loop system (the motor reference velocity $\omega_{Mi,r}(t)$, with Laplace transform $\Omega_{Mi,r}(s)$, $s \in \mathbb{C}$), to its output (the actual motor velocity $\omega_{Mi}(t)$). Furthermore, $\mathcal{I} = \{1, 2, 3\}$ represents the index set of sheet regions. Since the bandwidth of the low level control loops is required to be significantly higher than the bandwidth of the high level control loop [108], we can assume perfect tracking behavior of the controlled motors, i.e., $T_i(s) = 1$, $\forall i \in \mathcal{I}$.

Under the assumption of ideal behavior in the low level control loops, the inputs u_i of the high level sheet dynamics will be directly generated by the HLCs. This is shown in Figure 15.3, which represents the block diagram of the control system at hand. Since at each time instant the sheet is only driven by one pinch, the input of the sheet dynamics will change when the sheet arrives at the next pinch. This switching behavior can be easily captured in the piecewise linear (PWL) modeling formalism. The sheet velocity is derived from the motor velocities via straightforward holonomic kinematic constraint relations that describe the relation between motor velocity and pinch velocity, and pinch velocity and sheet velocity, respectively. The nominal high level sheet model, i.e. the sheet model without parameter uncertainties and disturbances, is:

$$\dot{x}_s = B_i \underline{u} \quad \text{for} \quad x_s \in \mathcal{X}_i, i \in \mathcal{I}, \tag{15.2}$$

with the input matrices B_i defined as $B_1 = \begin{bmatrix} n_1 r_{P1} & 0 & 0 \end{bmatrix}$, $B_2 = \begin{bmatrix} 0 & n_2 r_{P2} & 0 \end{bmatrix}$, and $B_3 = \begin{bmatrix} 0 & 0 & n_3 r_{P3} \end{bmatrix}$, respectively. In these definitions, n_i represents the transmission ratio between motor i and pinch i and r_{Pi} represents the radius of the driven roller of pinch i. Furthermore, $\underline{u}$ is the column with inputs of the high level sheet dynamics: $\underline{u} = \begin{bmatrix} \omega_{M1} & \omega_{M2} & \omega_{M3} \end{bmatrix}^T$. The partitioning of the state space into the three regions is represented by $\{\mathcal{X}_i\}_{i \in \mathcal{I}} \subseteq \mathbb{R}$. Here, $\mathcal{X}_1 = \{x_s | x_s \in [x_{P1}, x_{P2})\}$, $\mathcal{X}_2 = \{x_s | x_s \in [x_{P2}, x_{P3})\}$, and $\mathcal{X}_3 = \{x_s | x_s \in [x_{P3}, x_{P3} + L_s)\}$.

15.3 Control design and performance analysis

In this section, we present the controller synthesis method for the PWL sheet model (15.2), together with the performance analysis in case the paper path parameters are perturbed. Since we are dealing with a tracking problem, the system is formulated in terms of its tracking error dynamics. In contrast to the formulation of linear models in the error domain, the formulation of the PWL sheet flow model in error space yields a discontinuous model of the error dynamics. More specifically, the resulting model consists of both flow conditions, describing the dynamics in each regime, and jump conditions, describing the error dynamics at the switching boundaries. The flow conditions can be represented as follows [24]:

$$\dot{\underline{q}} = F\underline{q} + G_i\underline{\mu} \quad \text{for} \quad \left(x_{s,r} - \begin{bmatrix} 1 & 0 \end{bmatrix} \underline{q}\right) \in \mathcal{X}_i, i \in \mathcal{I}. \tag{15.3}$$

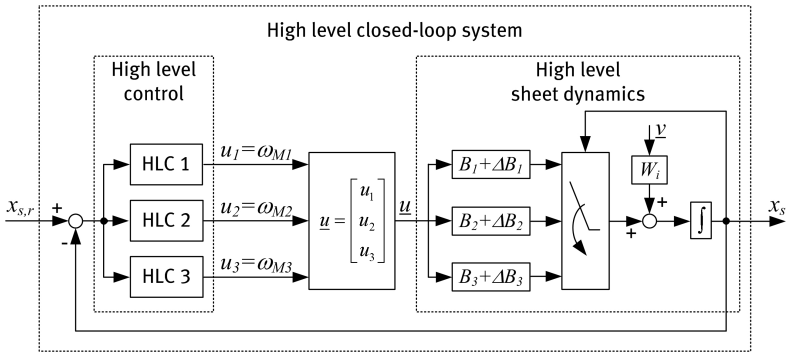


Figure 15.3: Block diagram of the total control system.

In this notation, the state vector $\underline{q}$ is defined as $\underline{q} = \begin{bmatrix} e_s & \dot{e}_s \end{bmatrix}^T$, with $e_s = x_{s,r} - x_s$ the sheet tracking error and the control input $\underline{\mu}$ is defined as $\underline{\mu} = \underline{\dot{u}}$. The system matrix is defined as $F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, whereas the input matrix is defined as $G_i = \begin{bmatrix} 0_{3\times 1} & -B_i^T \end{bmatrix}^T$. On the other hand, the jump conditions can be represented as

$$\underline{q}^+ = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{q}^- + \begin{bmatrix} \underline{0}^T \\ B_k - B_{k+1} \end{bmatrix} \underline{u}(t_s), k \in \mathcal{K}, \tag{15.4}$$

with $\underline{q}^-(t_s) := \begin{bmatrix} e_s(t_s^-) & \dot{e}_s(t_s^-) \end{bmatrix}^T$ and $\underline{q}^+(t_s) := \begin{bmatrix} e_s(t_s^+) & \dot{e}_s(t_s^+) \end{bmatrix}^T$ the state vector just before and after the switching time t_s, respectively. Furthermore, $\mathcal{K} = \{1, 2\}$ represents the index set indicating the possible switches between regime k and $k+1$. Hence, the complete model of the open-loop sheet dynamics in error space is given by the flow conditions (15.3) and the jump conditions (15.4).

Given this notation in error space, the controller synthesis can be carried out. For controlling the piecewise linear flow dynamics in error space (15.3) in combination with the jump conditions (15.4), we propose a control law that is based on state feedback of the error dynamics:

$$\underline{\mu} = -K\underline{q}. \tag{15.5}$$

Substitution of (15.5) into (15.3) yields the closed-loop flow dynamics in error space:

$$\dot{\underline{q}} = (F - G_i K)\,\underline{q} \quad \text{for} \quad \left(x_{s,r} - \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \underline{q}\right) \in \mathcal{X}_i, i \in \mathcal{I}. \tag{15.6}$$

The closed-loop jump conditions can be derived by substitution of the control law to be implemented, derived from (15.5), into the open-loop jump conditions (15.4), yielding

$$\begin{aligned} \underline{q}^+(t_s) \quad = \quad & \begin{bmatrix} 1 & 0 \\ B_{k+1}(k+1)K(k+1,2) - B_k(k)K(k,2) & 1 \end{bmatrix} \underline{q}^-(t_s) + \\ & + \begin{bmatrix} 0 \\ B_{k+1}(k+1)K(k+1,1) - B_k(k)K(k,1) \end{bmatrix} \int_{t_0}^{t_s} e_s(\tau)\mathrm{d}\tau, k \in \mathcal{K}. \end{aligned} \tag{15.7}$$

Given the total closed-loop error dynamics (15.6)-(15.7), Lyapunov-based stability analysis is combined with feedback controller synthesis for the PWL system at hand via the formulation of a set of Linear Matrix Inequalities (LMIs) that can be solved efficiently using commercially available software [50].

In case parameter uncertainties are present, the high level PWL sheet flow model becomes:

$$\dot{x}_s = (B_i + \Delta B_i)\,\underline{u} \quad \text{for} \quad x_s \in \mathcal{X}_i, i \in \mathcal{I}, \tag{15.8}$$

where ΔB_i is the constant uncertainty term of the i-th subsystem. In this model, this term can represent, for example, an uncertainty in the transmission ratio between motor i and pinch i or an uncertainty in the radius of the driven roller of pinch i. Based on this sheet flow model, the closed-loop flow and jump conditions can be derived in analogy with the derivation presented above. Based on the closed-loop jump conditions, the effect of the perturbations of the system parameters on the jumps in $\dot{e}_s$ can be predicted. As an example, the following holds for $\dot{e}(t_s^+)$ in case $e_s(t_s^-) = 0$ and $\dot{e}(t_s^-) = 0$,

$$\dot{e}_s(t_s^+) \quad = \quad \left(\frac{-B_k(k)\Delta B_{k+1}(k+1)+B_{k+1}(k+1)\Delta B_k(k)}{B_{k+1}(k+1)(B_k(k)+\Delta B_k(k))}\right)\dot{x}_{s,r}, k \in \mathcal{K}, \tag{15.9}$$

with $\dot{x}_{s,r}$ the sheet reference velocity. Hence, from (15.9) it can be concluded that the controller gains do not influence the jump in $\dot{e}_s$, since this jump is fully determined by the system parameters. For more detail, the reader is referred to [26].

15.4 Experimental setup

To experimentally validate the proposed control design approach, we use the paper path setup depicted in Figure 15.4. As can be seen in the figure, the setup consists of a PIM and a paper path with five pinches. In our experiments, only the second, third, and fourth pinch will be used. For the sake of notation, in the remainder of this chapter we will refer to these pinches as pinch 1, pinch 2 and pinch 3, respectively. Each pinch is connected to a motor via a gear belt. The nominal transmission ratios between the motors and pinches are $n_1 = 0.49$, $n_2 = 0.47$, and $n_3 = 0.5$, respectively, and the pinch radii are $14 \cdot 10^{-3}$ m. The motors are 10 W DC motors, driven by power amplifiers with built-in current controllers. The angular positions of the motor shafts are measured using optical incremental encoders with a resolution of 2000 increments per revolution. Both the amplifiers and the encoders are connected to a PC-based control system. This system consists of a Pentium 4 host computer running RTAI/Fusion Linux and Matlab/Simulink and three TUeDACS USB I/O devices [80]. The sheets are guided through the paper path via thin steel wires and their position is measured using optical mouse sensors, which are directly connected to the host computer via USB. Since these mouse sensors are incremental displacement sensors, they cannot be directly used for absolute sheet position measurement. However, using a dedicated calibration strategy in combination with an online data processing procedure, we can still use the mouse sensors in the sheet feedback control loop.

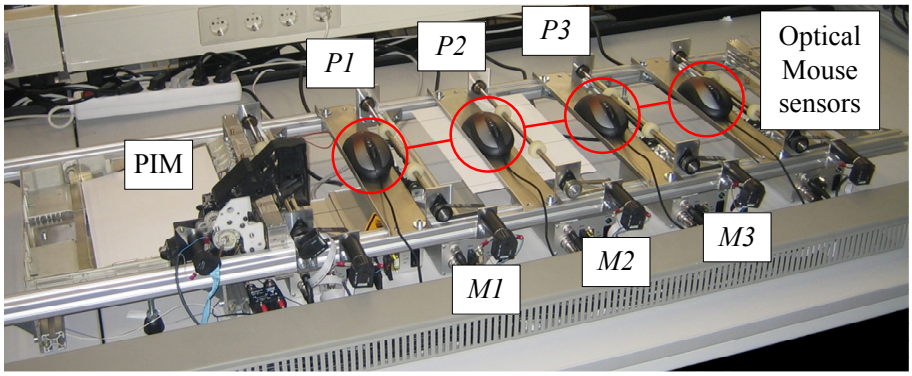


Figure 15.4: The experimental paper path setup.

15.5 Experimental results

15.5.1 Control design results

Based on the nominal values of the paper path parameters, presented in the previous section, the controller synthesis has been carried out, yielding the following controller gains:

$$K = 1 \cdot 10^5 \begin{bmatrix} -2.3 \cdot 10^5 & -9.9 \cdot 10^3 \\ -2.4 \cdot 10^5 & -1.0 \cdot 10^4 \\ -2.2 \cdot 10^5 & -9.6 \cdot 10^3 \end{bmatrix}. \tag{15.10}$$

To study the system performance in case of uncertain system parameters, the transmission ratios between the motors and pinches can be varied in the experimental setup. More specifically, the implemented transmission ratios are $n_1 = 0.49$, $n_2 = 0.53$, and $n_3 = 0.49$, i.e. the ratios of the second and third subsystem deviate from the nominal values. The performance analysis discussed in Section 15.3 showed that the jumps in $\dot{e}_s$ will be in the order of $4 \cdot 10^{-2}$ ms^{-1}. Since all subsystems are stable, the resulting sheet tracking error will be quickly controlled towards zero.

15.5.2 Low level motor control

In this subsection, the low level control of motor 1 and its influence on the high level sheet dynamics is discussed. Although not shown, similar results are obtained for motors 2 and 3.

In the design procedure of the sheet feedback controllers we assumed perfect tracking behavior of the controlled motors, i.e. we assumed an infinite bandwidth of the motor control loops. Furthermore, we assumed an infinitely stiff coupling between the pinches and the motors. In a practical environment, however, these assumptions do not hold. Moreover, a digital implementation will cause a delay in the loop which will limit the attainable bandwidth. Based on identified motor dynamics, PID feedback

controllers have been designed using loop shaping techniques [46]. The controller parameters are tuned such that a bandwidth of 50 Hz has been realized. This can be seen in Figure 15.6, which depicts the Frequency Response Function (FRF) of the loop gain. Here, the bandwidth is defined as the frequency at which the 0 dB line of the open-loop FRF is crossed.

The rubber belt that connects the motor with the driven roller of the pinch has a limited stiffness, as can be observed from Figure 15.5, which shows the FRF of the transmission between motor 1 and pinch 1. It can be seen that the assumption on the infinite stiff coupling between motor and pinch only holds for frequencies up to approximately 100 Hz. In this frequency range, the measured transmission ratio coincides with the nominal transmission ratio of 0.49 (≈ -6.3) dB. For higher frequencies, the flexibility becomes dominant.

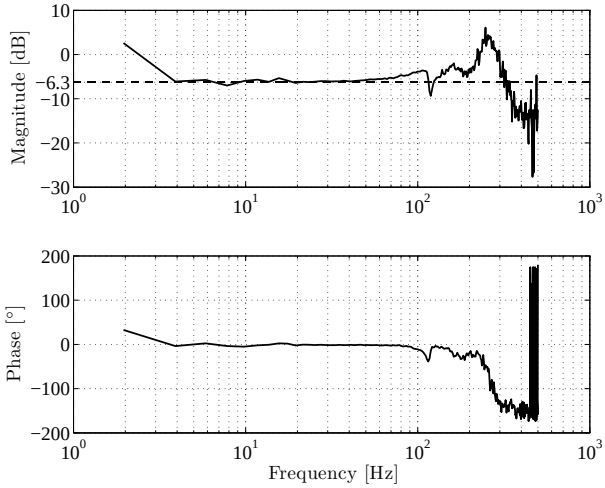


Figure 15.5: FRF of the transmission between motor 1 and pinch 1 (solid), and the nominal transmission ratio (dashed)

Given the high level sheet model and the HLCs, together with the controlled motor-pinch dynamics, the loop gain of the first subsystem can be derived. This loop gain is the transfer function from the sheet tracking error to the actual sheet position. The FRF of this loop gain is also shown in Figure 15.6. It can be seen that a bandwidth of approximately 10 Hz has been realized. This is a factor of 5 lower than the bandwidth of the motor control system, as required in a cascade control structure [108]. Furthermore, the phase lag at 10 Hz is approximately 90°. From this we can conclude that the first subsystem is stable.

In the control design procedure for the regulation of the PWL error dynamics, stability was proven for the case of perfect low level motor behavior. However, we want to apply the calculated controller (15.10) also in practical cases where we have to deal with non-ideal low level behavior, and still guarantee that the overall switched system

is stable. The stability of this switched system can be analyzed a posteriori by making a PWL model of the combined high level sheet dynamics and the controlled motor dynamics. After transformation of this model to the error domain and closing the loop using the HLCs (15.10), stability can be analyzed.

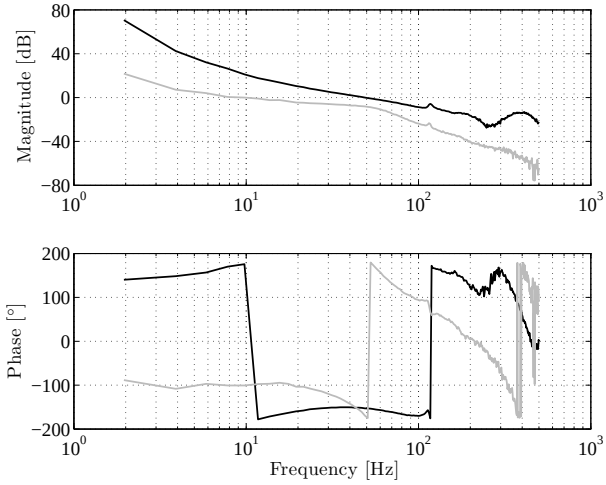


Figure 15.6: FRF of the loop gain of the first motor control loop (black) and the FRF of loop gain of the first subsystem, including low level control (gray).

15.5.3 Validation results

In the experimental validation of the control design, the focus is on the performance of the system in case of parameter perturbations. For the sheet motion task, a constant velocity of 0.3 ms^{-1} is chosen that has to be tracked throughout the entire paper path. The corresponding sheet reference motion $x_{s,r}$ is therefore a ramp function. Since no feed-forward control input has been used, all three pinches are standing still until a sheet enters the first pinch. Due to the difference between the initial reference velocity and the actual initial velocity, the sheet error starts increasing when the sheet enters the first pinch, as can be seen in Figure 15.7. However, this error is decreased quickly by the sheet controller in the first regime. Furthermore, it can be seen that the error increases when the sheet enters pinches two and three. Since the sheet tracking error is the input of the HLCs at all times, as can be seen in Figure 15.3, pinches two and three will already have a nonzero initial velocity when the sheet enters these pinches. Hence, the increase in tracking error is due to the deviation of the transmission ratios with respect to the nominal values. Also these increases are controlled towards zero quickly.

The response obtained from simulation is also depicted in Figure 15.7. It can be seen that there is a close match between the experimentally obtained sheet tracking

error and the one obtained in simulation. This close match justifies the assumption on ideal low-level motor dynamics in the controller synthesis approach.

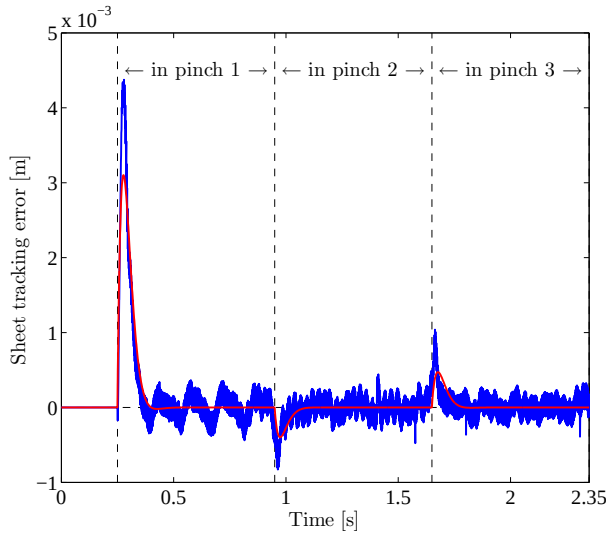


Figure 15.7: Experimentally obtained sheet tracking error (blue), together with the one obtained in simulation (red).

15.6 Conclusions and future work

In this chapter, a model-based control design approach for sheet feedback control in a printer paper path has been presented. Based on a simple sheet flow model with few details, sheet feedback controllers have been designed. The use of cheap optical mouse sensors as sheet position sensors has enabled the practical validation of the control design. Experiments show that a stable closed-loop system has been obtained, of which the responses can be predicted very well using the model. The approach opens an opportunity in industrial applications to use less expensive mechanics with larger tolerances, and still to achieve the desired printing quality. Current research focusses on the applicability of the approach in an industrial environment. Special attention will be on control design for cases in which pinches are coupled into sections, driven by one motor, and cases in which more than one pinch can influence the sheet motion. Preliminary results on these topics look promising, yielding the possibility to apply the approach on a realistic paper path as the one shown in Figure 15.1.

Chapter 16

Event-driven control

Authors: J.H. Sandee, W.P.M.H. Heemels and P.P.J. van den Bosch

16.1 Introduction

Control algorithms are indispensable for the proper functioning of many high-tech applications. For instance in a copier, where many controllers can be found controlling motors in the paper path driving the rollers, in the scanner driving an array of sensors to scan the media, or in the finisher, where paper trays are moved to the right position to catch the sheets of paper. Next to controlling motors, controllers are applied for various other purposes in the copier. One example is temperature control at the location where the image is fused onto the sheet. Also controllers can be found that are not controlling a physical element of the copier, but for instance take care of synchronized timing over the multiple processors in the system.

These control algorithms are typically executed on a real-time software processing platform, under strong real-time conditions to guarantee their required control performance. The major cause of these imposed conditions is that most controller design methods are based on the requirement that the controller sample moments are uniformly distributed over time, i.e. having fixed sample intervals. As a consequence, control engineers pose strong, non-negotiable requirements on the real-time implementations of their algorithms. This is illustrated in Figure 16.1, which depicts the control algorithm as a package that is thrown over a brick wall to the software department that has to implement and test the algorithm.

There are various issues that make the implementation of controllers difficult on embedded platforms with limited resources. These issues result in important *trade-offs* that affect both the control performance (i.e. tracking, stabilization, disturbance rejection, et cetera) and the software performance (i.e. processor load, response times, et cetera) and have to be dealt with in the system design. Typical trade-offs are for in-

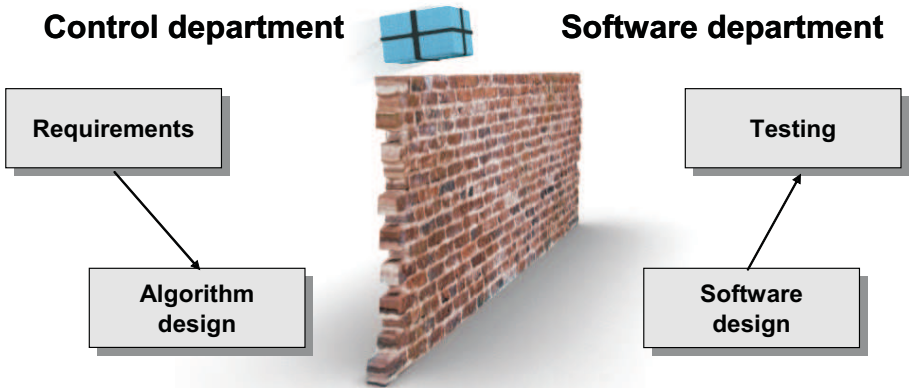


Figure 16.1: ...control engineers pose strong, non-negotiable requirements on the real-time implementation of their algorithms...

stance found in the selection of the sample frequency for the control algorithms. The higher the sample frequency is chosen, the better the control performance generally is that can be obtained. But when increasing the sample frequency, the processor load rises, for the simple reason that it takes time to execute the control algorithm computations. High processor loads can be a real problem in applications where processing power is limited. Another typical trade-off is found in quantization. Quantization is often caused by the limited resolution of sensors, but also the software implementation and communication mechanisms can be important reasons. When a controller is implemented on a specific platform we have to deal with a limited resolution for the representation of variables. This is caused by the limited word length. Depending on the processor, calculations will cost more time for bigger word lengths. Therefore, increasing the word length is an advantage for the control performance because of reduced quantization, but might as well be a disadvantage because of the increased computation time. For communicating data over a network with limited capacity, the same reasoning applies.

When considering the trade-offs between software and control engineering, an obvious step forward would be to have design methods for control algorithms that take the requirements of the software implementation into account. Researchers in software and control engineering are becoming increasingly aware of this need for an integrated scientific and technological perspective on the role that computers play in control systems and that control can play in computer systems [102]. The research presented in this chapter focusses on reducing the gap between both disciplines (Figure 16.2), by relaxing one of the most stringent conditions that control engineers impose: a fixed

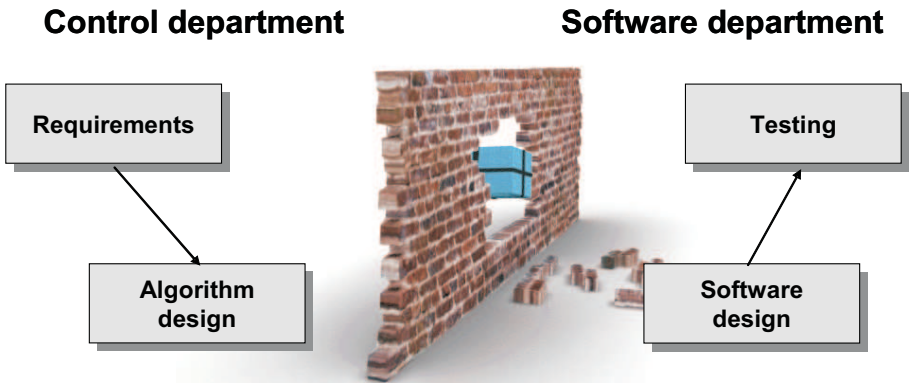


Figure 16.2: ...a step towards reducing the gap between software and control engineering...

sample frequency.

We propose control algorithms that do not require that sample moments are uniformly distributed over time. We claim that this enables the engineers to make better trade-offs in order to achieve a better overall *system performance*. By not requiring equidistant sampling, one could for instance vary the sample frequency over time and therefore choose to dynamically schedule the control algorithms in order to optimize over processor load. Another option is to design the controller such that it responds faster to acquired measurement data with which quantization effects and latencies are reduced considerably.

These controllers we call *event-driven* controllers, as it is an *event*, rather than the progression of time, that triggers the controller to perform an update. The classical controllers that perform equidistant sampling we call *time-driven*. For time-driven controllers it is the autonomous progression of time that triggers the execution of actions.

Research hypothesis

> Event-driven control is an improvement over traditional time-driven control to achieve a better overall *system performance*, by relaxing one of the most stringent conditions that control engineers impose: a fixed sample frequency.

The work presented in this chapter is part of the published PhD-thesis: 'Event-driven control in theory and practice - trade-offs in software and control performance' [100].

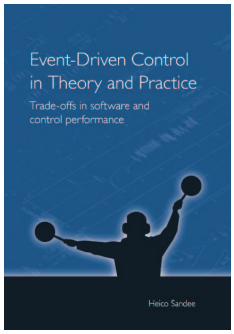


Figure 16.3: Thesis cover of [100].

16.2 Event-driven control

To illustrate the difference between time-driven control and event-driven control, take the example of a mailman delivering packages to customers [66]. In the time-driven situation every customer uses the wall clock to check the door for a new package every 5 minutes. When no package has arrived, they can resume their work. In the event-driven situation the mailman rings the doorbell of the specific customer who he has to deliver a package. This customer opens the door and accepts the package. The other customers can continue their work without being interrupted. This example clearly illustrates one of the possible benefits of event-driven control, which is a reduction of the work load, as customers do not have to open their doors unnecessarily. In a control application this is translated to a reduction of, for instance, the communication bus load and processor usage. From a control *performance* point of view, the real advantage of event-driven control is the reduced response times. When a package is delivered, the customer is alerted and can open the door immediately. In the time-driven situation it may take up to 5 minutes after delivery until customers take action to it.

The aim of event-driven control is to create a balance between the control performance and other system aspects. Event-driven control can for instance reduce the processor load, while maintaining a high control accuracy, by only computing control updates when the measured signal deviates significantly from the reference signal, or only when new measurement data comes available. Also sensor resolutions can be reduced considerably, by designing the controller such that it specifically deals with the event-based nature of the sensor. These reduced sensor resolutions have clear cost price advantages.

In the thesis [100], two event-driven control schemes are presented, that have been successfully applied in the printer case study. The first one uses an (extremely) low resolution encoder to measure the angular position of a motor. The event-driven controller is designed such that actuation is performed right after the detection of an encoder pulse. In this way, the controller can use the *exact* position measurement, and

is not affected by the quantization errors of the encoder. Moreover, the controller can respond fast to measurement data. When the motor is not running at constant velocity, the updates are not equidistant in time. It is therefore not possible to use the classical design methods which all assume that updates are equally spaced in time. We can however apply variants of classical design methods if we define our models of the plant and the controller in the (angular) position domain instead of the time domain, as proposed in [55]. This idea is based on the observation that the encoder pulses arrive equally spaced in the position (spatial) domain. By applying this event-driven controller, we not only decrease the encoder resolution - and therefore the system cost price - but also the average processor load, compared to the conventional controller. This was accomplished without degrading the control performance, with respect to the originally applied controller.

The focus of the second type of event-driven controller is to obtain a high control performance on the one hand and realizing a reduction of the resource utilization (processor load, communication bus load) on the other. This is realized by updating the controller only when the (tracking or stabilization) error is larger than a threshold and holding the control value if the error is small. Already in 1962, the need for such controllers was addressed [37], but few research has been spent in this subject since. We aim particularly at a mathematical analysis of such controllers to start building an event-based system theory. The proposed controller is furthermore experimentally validated to research the real benefit in terms of processor load reduction and not only the number of control updates.

In [100] event-driven control is explained in more detail with various theoretical and practical examples. In this chapter we will consider one particular example of event-driven control in more detail, as applied in the printer case study. This example is based on the first proposed event-driven control scheme to accurately control a motor based on a low resolution encoder.

16.3 Sensor-based event-driven control

To keep the system cost price limited, our aim is to use low resolution encoders for the control of motors. However, now the quantization errors become significant when applying time-driven control and are not negligible anymore. To still achieve satisfactory control performance, this requires an adjustment to conventional control algorithms to deal with this low-resolution encoder signal, as these algorithms assume continuous sensor read-out.

Most applied and researched solutions that deal with noisy and low resolution sensor data use an observer-based approach to estimate the data at *synchronous* controller sample moments, based on *asynchronous* measurement moments [13, 22, 49, 53, 72, 93, 76, 119]. In these solutions, the continuous-time plant is translated into a discrete-time model which is time-varying, depending on the time between successive measurement instants.

A completely different approach has been taken in [55], in which a simple control

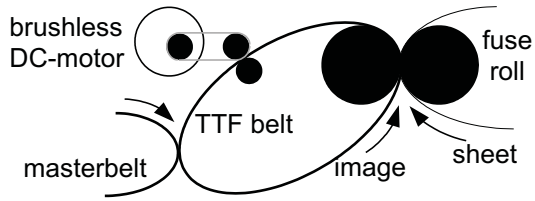


Figure 16.4: Schematic representation of the printing process.

structure is presented for the control of a slave motor in master-slave combinations, that does not suffer from the added complexity of an observer. The control structure is an asynchronous control scheme in which the control updates are triggered by the slave position measurement (encoder pulse). The idea of the asynchronous controller is based upon the observation that at an encoder pulse the position is *exactly* known and thus there is no need for an observer as in the before mentioned approaches. However, as the velocity of the motors vary over time, both measurement and control updates are not equidistant in time. This requires a completely new design paradigm for these event-driven controllers. We have applied a similar controller structure as proposed in [55] and extended the controller analysis and design techniques to accurately control a brush-less DC-motor in the printer that drives the TTF-belt to control the motion of images through the printer (Figure 16.4), on the basis of a very low resolution Hall encoder.

The brush-less DC-motor that is driving the TTF-belt is modeled by the second-order model

$$\begin{aligned} \dot{\theta}(t) &= \omega(t) \\ \dot{\omega}(t) &= \tfrac{1}{J}[(\tfrac{-k^2}{R} - B)\omega(t) + \tfrac{k}{R}u(t) - d(t)] \end{aligned} \tag{16.1}$$

where $\theta(t)$ $[rad]$ is the angular position of the motor axis, ω(t) is its angular velocity $[rad/s]$, $u(t)$ the motor voltage $[V]$ and $d(t)$ the disturbance torque $[Nm]$ at time $t \in \mathbb{R}$. The motor parameters are obtained from data sheets of the motor manufacturer: the motor inertia (including the load inertia) $J = 1.83 \cdot 10^{-4}\ kgm^2$, the motor torque constant $k = 0.028\ Nm/A$, the motor resistance $R = 1.0\ \Omega$ and the motor damping $B = 3.0 \cdot 10^{-5}\ Nms/rad$.

As presented in [100] we can apply variants of classical design methods, if we define our models of the plant and the controller in the spatial (angular position) domain instead of the time domain. This idea is based on the observation that the Hall pulses arrive equally spaced in the spatial domain, as the Hall sensors have an equidistant distribution along the axis of the motor. To use this reasoning, we first have to transform the motor model as given in Equation (16.1) to an equivalent model in which the motor angular position is the independent variable. After that, the controller design can be performed using classical control theory.

The motor model is transformed to the spatial domain via the following relation:

$$\frac{d\theta}{dt}(t) = \omega(t) \Rightarrow \frac{dt}{d\theta}(\theta) = \frac{1}{\omega(\theta)}, \tag{16.2}$$

where $\omega(\theta)$ denotes the angular velocity of the motor and $t(\theta)$ denotes the time, respectively, at which the motor reaches position θ. Under the assumption that $\omega(t) \neq 0$ for all $t > 0$, a one-to-one correspondence between θ and t exists and an interchange of their roles is possible. Note that $\omega(t) \neq 0$ is valid under normal operating conditions for the considered example, as the motor does not change direction.

Using (16.2) we obtain the motor model in the spatial domain:

$$\begin{array}{l} \frac{dt}{d\theta}(\theta) = \frac{1}{\omega(\theta)} \\ \frac{d\omega}{d\theta}(\theta) = \frac{1}{J}[-\frac{d(\theta)}{\omega(\theta)} - (\frac{k^2}{R} + B) + \frac{k}{R} \cdot \frac{u(\theta)}{\omega(\theta)}] \\ y(\theta) = t(\theta) \end{array} \tag{16.3}$$

where $d(\theta)$ and $u(\theta)$ denote the disturbance torque and the motor voltage, respectively, at motor position θ. Interestingly, the output $y(\theta)$ of this new representation is now the time $t(\theta)$ at which the motor reaches position θ. To consider the disturbance d as a function of the angular position θ is an advantage for many controller designs, as disturbances are often coupled to the angular position, instead of time. Examples can be found in bearings, axes, rolls, traveling sheets of paper, et cetera, that all rotate at a multiple of the velocity of the motor. When the motor velocity decreases, all frequencies of the disturbances decrease with the same factor.

We can now apply classical control design methods to design a controller for this non-linear plant model. The details of the procedure can be found in [100]. The analysis of the control performance is carried out in the spatial domain. As a result, the measures that we obtain from analysis are also in this spatial domain. For instance the bandwidth of the controller, which is defined as the frequency up to which disturbances are sufficiently suppressed, is not expressed in Hz anymore, but in rad^{-1}. Furthermore, we do not aim at obtaining a certain settling time but settling *distance*. These measures can give very valuable information about the control performance, as these spatial measures directly couple to the print quality.

To validate the controller analysis and synthesis, we compared the designed event-driven controller with the originally used hybrid controller. In the hybrid control scheme the actuator signal is updated at a constant rate (i.e. synchronous in time) but measurements are done asynchronously in time. Each moment a new Hall pulse is detected, a time-stamp is taken. At each synchronous control update, this time-stamp is used to estimate ω and θ at the control update times from the asynchronous measurements. Both the event-driven controller and the hybrid controller are implemented on a complete prototype document printing system (as the one shown in Figure 1.6). The motor model (16.3) was matched with this prototype system. Therefore, the controller parameters obtained from the analysis could be applied *directly* to control the TTF belt in the prototype.

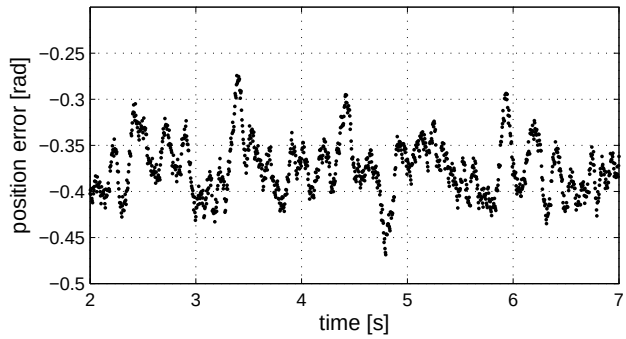


Figure 16.5: Experiment hybrid controller with 12 PPR encoder and sample freq. 250 Hz.

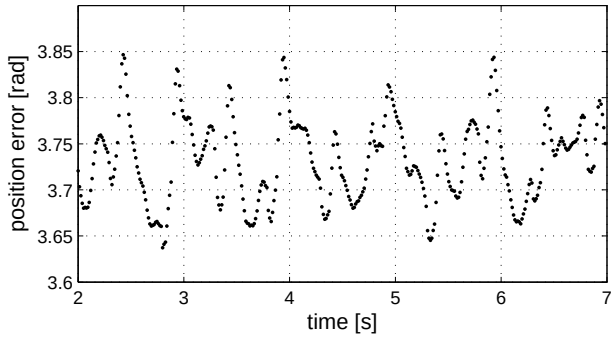


Figure 16.6: Experiment event-driven controller [100, Eq. 4.22] with 1 PPR encoder.

The experimental results for both controllers are given in Figures 16.5 and 16.6. These figures show the position error during printing over 5 seconds (after start-up). In this period, 5 sheets are printed at a speed of 80 pages per minute. Comparing the results in Figures 16.5 and 16.6 we observe similar control performance for the hybrid controller and the event-driven controller. For the control performance we mainly consider the deviation from a steady-state position error during printing, as only deviations from a constant position error will be visible in the print quality. From the figure we see that the maximum deviation from a constant position error varies for both controllers within a range of ± 0.15 rad, as was required. However, keep in mind that the event-driven controller operates with an encoder with a resolution that is a factor 12 lower than used for the hybrid controller. Furthermore, the hybrid controller runs at a (constant) control sample frequency of 250 Hz and the event-driven controller at a much lower *average* frequency (approximately 62 Hz). The errors caused by the sheet passings can be distinguished in both figures, although there are more disturbances (at different frequencies) acting on the system as can be seen from the measurement data.

Comparing the processor load for both controller algorithms, it is shown in [100] that the event-driven controller reduces the load with a factor 6 compared to the hybrid controller implementation, for the considered situation.

16.4 Conclusions

Event-driven control is presented in the research hypothesis as a control design method to achieve a better overall system performance, compared to classical time-driven approaches, by relaxing one of the most stringent conditions that control engineers impose: a fixed high sample frequency. System performance has to be understood in the sense of the combination of aspects that are influenced by the controller implementation. These are in particular: control performance (in terms of tracking, stabilization and disturbance rejection), software performance (in terms of processor load), amongst other aspects like communication bus load and system cost price.

In this chapter we considered one particular event-driven controller, which shows that by relaxing the equidistant sampling constraint, event-driven controllers can respond faster to changing conditions. The update of the proposed controller is triggered by new sensor data that comes available, which are the individual pulses of an encoder in the considered case. This means that the *exact* position measurement is used, instead of some estimation with a non-zero measurement error, which opens up the possibility to achieve high control performance, while operating with cheap, low resolution sensors. The controller tuning, for this fundamentally different controller compared to classical time-driven controllers, was performed by transforming the system equations from the time domain to the spatial domain. In the spatial domain, the encoder pulses, and therefore the controller triggering, occur equidistantly spaced. In this way, we are able to write the control problem as a synchronous problem such that classical control theory can be applied to design and tune the controller. The resulting control performance measures are also expressed in the spatial domain, such that we obtain

the bandwidth in the spatial frequency and aim at settling distances instead of (classical) settling times. When disturbances are also acting in the spatial domain - which is often the case - it can easily be determined how these disturbances are rejected. The proposed event-driven controller was experimentally validated in the printer where a one pulse per revolution encoder is used to accurately control the motion of images through the printer in the case study. By means of experiments on a prototype printer we have shown that with the event-driven controller a similar control performance can be achieved, compared to the originally applied hybrid controller in combination with a 12 pulse per revolution encoder. Furthermore, it was investigated that the processor load for the controller was reduced up to a factor 6.

Chapter 17

Design trajectory and controller-plant interaction

Authors: P.M. Visser, J.F. Broenink and J. van Amerongen

17.1 Introduction

In the design process of a system various simulations and experiments should be performed to study the behavior of the system, to analyse the system performance and to make design decisions. As virtually all high-tech systems are multi-disciplinary in nature the modeling and simulation methods must deal with interaction between the various disciplines to support the system design. It is important to bring the disciplines together in an early design stage in order to avoid severe problems at the system integration phase that cause delays and additional design effort (see the Boderc research hypothesis in Chapter 1).

In this chapter a system design trajectory is proposed that facilitates the design steps from initial models to final realization. The system (or part of the system) is considered to be composed of three components: the plant, the input/output (I/O) interface and the controller as depicted in Figure 17.1.

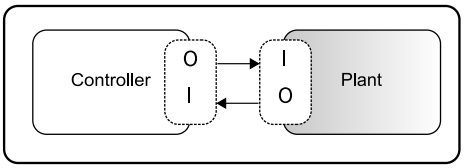


Figure 17.1: System scope

The plant is a physical device that can be controlled via the input/output (I/O) interface. For example, the paper path or a single pinch of the paper path that is driven by a motor. For the controller we will only consider the feedback control part, which in most modern systems is realized in software and embedded in the complete software of the system. The reason to focus is on the feedback control instead of the complete software is that the feedback control dominates the requirements of the hardware and software architecture. The emphasis of the feedback controller in this chapter is on the implementation on the control computer from the point of view of the software discipline; the control algorithm is supposed to be given.

Figure 17.2 shows an example of such a controller-plant system. The plant considered here is a motor that drives a pinch. The controlled variable of the plant is the angular position of the motor which can be measured by an encoder. The value of the encoder is sampled and forms the input of the digital feedback controller. The calculated output of the controller is applied by means of pulse width modulation (PWM) to the plant.

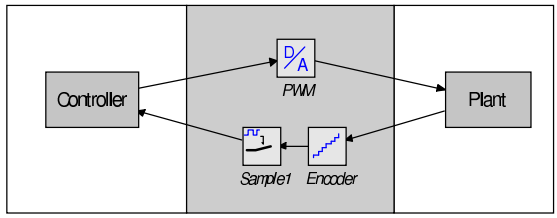


Figure 17.2: Controller-plant overview

The starting point of our system design trajectory consists of a running simulation of a plant model, a digital feedback controller model and the corresponding I/O. The plant does not have to be prototyped yet, the design trajectory can be commenced in the early design phases.

The final realization consists of the physical plant and the digital feedback controller that runs on the target. A target is a computer that is capable of computing the control law of the feedback controller.

The expected benefit from using the proposed design trajectory for the industrial user is a substantial reduction of the design time due to a reduction in integration effort. Moreover, as the interaction between the disciplines is clearer from the start of the design process, better choices can be made to improve the overall system behavior.

17.2 System design trajectory

The design trajectory depicted in Figure 17.3 proposes a systematic stepwise design trajectory with the goal to obtain a less error-prone path from model to realization [118]. It is a model-driven approach in which simulations are used to check whether refine-

ment updates keep the model compliant with the requirements. Via various ‘in-the-loop simulations’, the design trajectory runs from complete simulation (stage 1) to final realization (stage 6). By dividing the design in multiple stages, possible errors are isolated and can be diagnosed faster. In each stage a verification test is performed. If a verification test fails one should locate and solve the error in the refinements with respect to the previous stage and repeat the verification test.

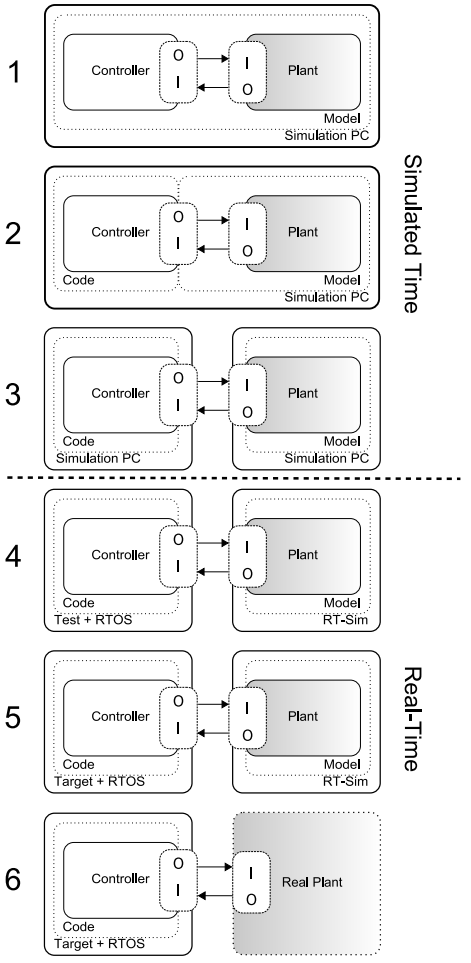


Figure 17.3: System design trajectory

In Figure 17.3 the simulated-time domain (stage 1, 2 and 3) and the real-time domain (stage 4, 5 and 6) are separated by a dashed line. The arrows between the I/O boxes denote the interconnections, which in the final realization stage are the connecting cables. The three components (controller, plant and I/O) are surrounded by a dotted

box which is the realization 'form' of the components. The realization form can be the model, the code or the real instantiation (the physical plant). The outer solid box is the platform on which the model or code is simulated/executed. A platform is a computer that is capable of performing the calculations required to perform the simulation or execution. The following platforms are used:

- Simulation PC platform, a PC that is capable to execute the model simulation.
- Real-Time Simulation PC platform, a fast PC with a real-time simulator, that is capable to simulate the plant model in real-time.
- Test platform, any commercial off-the-shelf (COTS) platform that can be used to run the controller.
- Target platform, the platform that is used in the final realization.

The design trajectory will be explained per stage.

Stage 1

In this stage both the controller and plant are simulated in the same modeling environment/tool on the same Simulation PC. The plant model will be simulated with a numerical integration method to approximate the continuous-time behavior. In order to obtain a deterministic computation time a fixed step-size numerical integration method is chosen. This is to ensure that the plant can be simulated in real-time (stages 4 and 5). The step-size of the numerical integration method has to be chosen, among others, to adequately handle the plant dynamics [19] (see also Section 17.3.3).

The simulation will be used to analyze the plant behavior and optimize the controller. This stage can be called Model-In-the-Loop Simulation.

Stage 2

In this stage the controller model has been transformed to executable code by means of code generation, also called synthesis export, and compilation. The controller executable runs simultaneously with the simulation of the plant model. The verification test is obvious: the simulation results here should be identical to the simulation results of stage 1.

In principle, no discrepancies are expected, because the control-laws executed in simulation and executed in the executable should behave identical. An error that could occur, for example, is that a different floating point library is used in the controller model and the controller executable.

The purpose of this stage is to check that generated code yields exactly the same results as the simulation in stage 1, i.e. that the code generation from the modeling tool and compiler works as expected.

Stage 3

In this stage the controller executable and the plant model run on two separate Simulation PC's. The controller executable runs simultaneously with the simulation of the plant model. The interconnection between the Simulation PC's is via digital I/O. This implies that the I/O signals are still numbers and not yet physical signals. The controller runs in a non-real-time environment as a task.

This stage is used to obtain a rudimentary estimation of the CPU usage, which can be used to facilitate the choice for the target hardware. The simulation results should be identical to stage 2.

Stage 4

In this stage both the plant and controller run in *real-time* on two separate computers: the Test platform for the controller and the RT Simulation PC for the plant. The real-time simulation of the plant model must resemble the real plant behavior closely. Hence, the simulated plant model must have the same interface as the real plant, requiring that the I/O signals are the real physical signals. The simulated plant must be replaceable by the real plant without any modifications of the controller.

By choosing a Test platform similar to the Simulation PC's in the previous stages, only the migration of the nature of time is verified here. Since in stage 1, a fixed step-size numerical integration method was chosen, both pieces of code generated from the model (controller and plant) are functionally identical to stage 1 and should yield the same simulation results. However, due to the real-time setting, no synchronized communication between the plant and controller (which is explained in detail in Section 17.3.3) and limited computation time, simulation results may differ compared to those in the previous stages.

In this stage the real-time behavior of the controller can be studied and accurate processor and memory usage can be determined in order to determine the target platform. For example, a design trade-off can be made to accept a worse control performance (e.g. by changing the sample frequency) or chose for a faster (more expensive) target. Depending on the trade-off one should return to a previous stage to analyse the effect.

Although the target platform is not necessarily used for the controller, this stage can be considered as Hardware-in-the-loop-Simulation since various COTS Test platforms can be used to support the selection of the final target platform. The constraints posed by the target system are dealt with in the next stage.

Stage 5

In this stage the target platform replaces the test platform at the controller side. The transformation to this stage may be complicated by specific compilers and/or hardware resource limitations. Hence, the transformation to this stage will consume more time and should be taken after the hard real-time behavior is analyzed in the previous stage.

The verification test should show similar behavior compared to the previous stage. Similar but not identical since the timing of the target system and the compilers used may differ from the test platform. If the verification test is successful the real plant can be connected.

Stage 6, the realization

In this final stage, the plant model is replaced by the real plant. Because in stage 4 the interface of the real-time simulation of the plant model was similar to the real plant only the cable-ends of the target system need to be connected to the real plant. The controller and the target system are the same as in the previous stage.

Differences in the simulation results compared to the Hardware-in-the-loop-Simulation (stage 5) are caused by the difference between the plant simulation and the real plant.

In this stage the behavior of the final system should satisfy the requirements. If the requirements are met the design process has been successfully performed. If the requirements are not met, one should return to a previous stages to solve the issue.

A case study in [118] illustrates the results of the design trajectory.

17.3 Controller-plant interaction

This section will illustrate that the choice of the controller implementation approach is not strictly an implementation issue but a design issue which can have a large impact on the overall performance of the system and influences many design disciplines.

The controller-plant interaction is explained by using diagrams. In order to keep the diagrams readable and uncluttered the following simplifications and notational conventions are used:

- The controller does not receive an event directly at the time at which it occurs but receives the event at the next sampling moment. In implementation terms, the I/O buffers the event until the controller reads the buffer.

- The plant is simulated with a fixed step-size of T_c. $T_c(k_c)$ is used to denote the time of the simulated plant model at time $t = k_c T_c$.

- The sampling interval of the controller is T_d. $T_d(k_d)$ is used to denote the time of the digital controller at time $t = k_d T_d$.

- The step size of the plant simulation is chosen ten times smaller than the sampling interval of the digital feedback controller (T_d=$10T_c$). This choice is explained in Section 17.3.3.

17.3.1 Controller implementation approaches

The approach that is most common in practice is time-driven control. In time-driven control there are two different control approaches [7, pages 328-330], which are depicted in Figure 17.4. The values T_{AD} and T_{DA} are the analog to digital and digital to analog conversion times, which are assumed to be constant. The value T_{Comp} is the computational time required to compute the control signal. The computational time will vary and is denoted with $T_{Comp}(k_d)$.

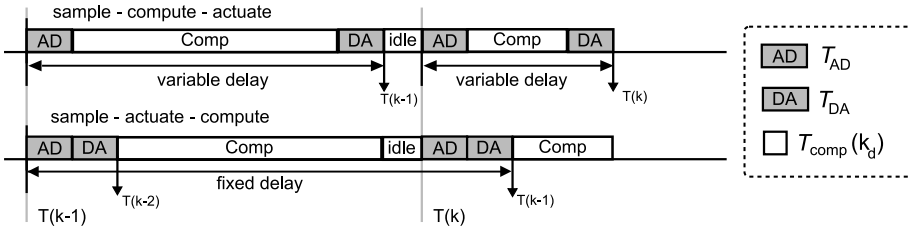


Figure 17.4: Control implementation approaches

The behavior of both approaches is as follows:

- 'sample-compute-actuate': a sample is taken on time $t = T_d(k_d)$ and used to compute the control signal which is applied on $t = T_d(k_d)+T_{AD}+T_{Comp}(k_d)+T_{DA}$. In this approach the control signal is applied with a varying time delay since the computational time could vary. The time between two control updates is periodic with jitter $(T_d + T_{Comp}(k_d + 1) - T_{Comp}(k_d))$.
- 'sample-actuate-compute': the control actuate signal for $t = T_d(k_d)$ is computed with the sample taken at $t = T_d(k_d - 1)$ and applied at $T_d(k_d) + T_{AD} + T_{DA}$. In this approach the control signal is applied with a fixed time delay equal to the sampling interval (T_d) plus the conversion times ($T_{AD} + T_{DA}$). The time between two control updates is periodic (T_d).

In both approaches there is a time delay between the moment a sample is taken and the moment the control signal is applied. To avoid issues in the final realization (stage 6) the time delay should explicitly be taken into account at the start of the design (stage 1). Therefore, the example controller-plant system of Figure 17.2 is extended with the addition of a time delay depicted in Figure 17.5 to deal with the control implementation approach. In the sample-compute-actuate approach, the time delay is $T_{Comp}(k_d)$. In the sample-actuate-compute approach, the time delay is T_d.

17.3.2 Interaction in a simulated-time simulation

The interaction in case of simulated-time simulation (stage 1, 2 and 3) is depicted in Figure 17.6. On the left side the sample-compute-actuate interaction is depicted and on the right side the sample-actuate-compute interaction is depicted.

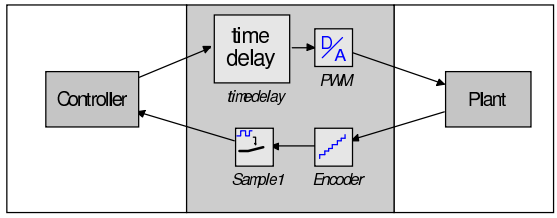


Figure 17.5: Controller-plant system with time delay

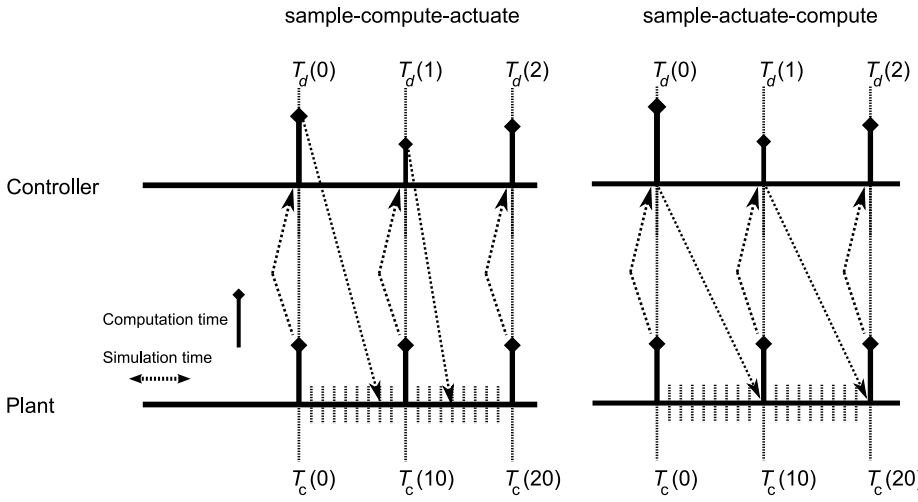


Figure 17.6: Simulated-time interaction

The main observation is that the control signal for a signal sampled at time $T_c(0)$ will be applied at $T_c(1) < t < T_c(10)$ in case of sample-compute-actuate (varying delay) and exactly at $T_c(10)$ in case of sample-actuate-compute (fixed delay).

In the simulated-time both the controller and the plant are synchronized by the modeling environment which prevents the occurrence of drift ($T_d(1) = T_c(10)$).

To accurately model the varying time delay (caused by using a computer to implement the control law) in the sample-compute-actuate approach, one needs detailed knowledge about the final target on which the controller will run. The delay can be estimated by analyzing the time that the instructions of the control algorithm (e.g. PID) will take on the target. A maximum time may be chosen if it can be determined that the varying delay does not hamper the control performance. Simulation can be used to study the impact of the varying delay.

In case of the sample-actuate-compute approach the delay is fixed to a unit delay (T_d). This requires no knowledge of the target and a proper controller design is able to

deal with such a fixed time delay.

From a system point of view the sample-actuate-compute approach is preferred. The approach allows a predictable design since no knowledge is required of the target, which may not be chosen at the start of the design process. In systems were the 'best' obtainable control performance is required and costs are of secondary importance, the sample-compute-actuate approach may be preferable. In such a system the varying delay will be small with respect to the sampling interval (T_d), since a high performance computer is used for computing the control algorithm.

For both approaches the target has to be chosen fast enough to compute the control algorithm in time.

17.3.3 Interaction in a real-time simulation

The interaction in case of real-time simulation (stage 4 and 5) is shown in Figure 17.7. On the left side the sample-compute-actuate interaction is depicted and on the right side the sample-actuate-compute interaction is depicted. The computation time is strictly coupled with real-time, denoted with squares.

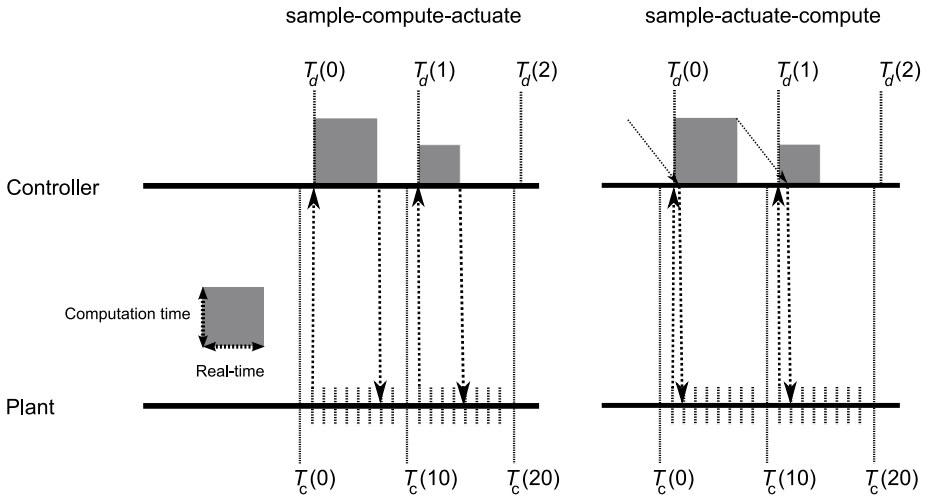


Figure 17.7: Real-time interaction

In real-time simulation there is no synchronization of time between the simulated plant model and the controller. Opposed to the *simulated-time* simulation, the controller will not wait for the plant to calculate its output and vice versa. In the simulated-time domain computing a second in simulation may take 10 seconds of computation time. In the real-time domain computing a second in real-time simulation must take less (or equal) than a second. As a consequence of the asynchronous behavior the time may drift ($T_d(1) \neq T_c(10)$). The asynchronous behavior is caused by fact that the controller

and the simulation of the plant model run on separate computers with separated clocks. The impact on the simulation results caused by this asynchronous interaction depends on the clock drift and the step-size of the plant simulation. Clock synchronization e.g. by hard-wiring is not 'allowed' because the idea of Hardware-in-the-Loop simulation is that the simulated plant can be replaced with the real plant without any modifications. The error caused by this asynchronous interaction is at least one numerical integration step (T_c). Hence, decreasing the step-size (T_c) of the plant simulation will decrease the error caused by the asynchronous interaction. A ratio of at least $10T_c \geq T_d$ is advised.

The real-time simulation results for the sample-actuate-compute approach will be similar to the simulated-time simulation results if the target is fast enough to compute the controller output in time. The results will not be identical because of the asynchronous interaction.

The simulation results for the sample-compute-actuate approach may differ from the simulated-time simulations. This is the case when the estimated time for the computational delay (used in stages 1,2 and 3) is not the same as the actual time that is required for the computation in the real-time simulations. If the control performance is not satisfactory, one has to adjust the estimated computational time in one of the previous simulated-time stages.

17.4 Conclusions and discussion

In this chapter we presented a systematic stepwise design trajectory to obtain a less error-prone path from model to final realization. For two common control implementation approaches the controller-plant interaction was discussed and indicated how they should be handled within the design trajectory.

In the design trajectory simplifying assumptions were made on the controller-plant interaction. Future work will focus on removing these assumptions. In particular, in the controller-plant interaction the events were only received by the controller at the sample moments. This is a limitation as many reactive systems need to deal with events at the moment they occur. Hence, the next step is to deal with events, both in the simulated-time simulations and the real-time simulations, in a realistic manner. When events can be taken into account the stepwise refinement trajectory can be extended from time-driven (synchronous) control to event-driven (asynchronous) control as discussed in Chapter 16.

Chapter 18

Impact, lessons learned and conclusions

Authors: G.J. Muller and W.P.M.H. Heemels

18.1 Introduction

The Boderc project started in 2002 and ended in 2006, implying that a total of 5 years of research is represented by the findings of this book. In this last chapter, we will look at four different aspects, namely

- the summarized project results,
- the research approach: industry-as-laboratory,
- the lessons learned in process and organization,
- the impact and spin-off,

after which we present the final conclusions of the book.

18.2 Project results

The previous chapters described the specific research outcomes of the Boderc project. In this section we summarize the results and position them in the design pyramid, see Figure 18.1. It shows that the ordering of the Boderc symposium book is top-down. Moreover, it indicates that the results are reasonably well distributed over the different abstraction levels. A first rough ordering was already indicated in Figure 1.9. Most

PhD-theses are connected to the existing scientific body of knowledge, a level of detail that goes beyond the bottom of the pyramid due to the current academic standards. However, the continuous pull towards multi-disciplinary knowledge has resulted in several theses that range from detailed scientific up to a certain level of multi-disciplinary design.

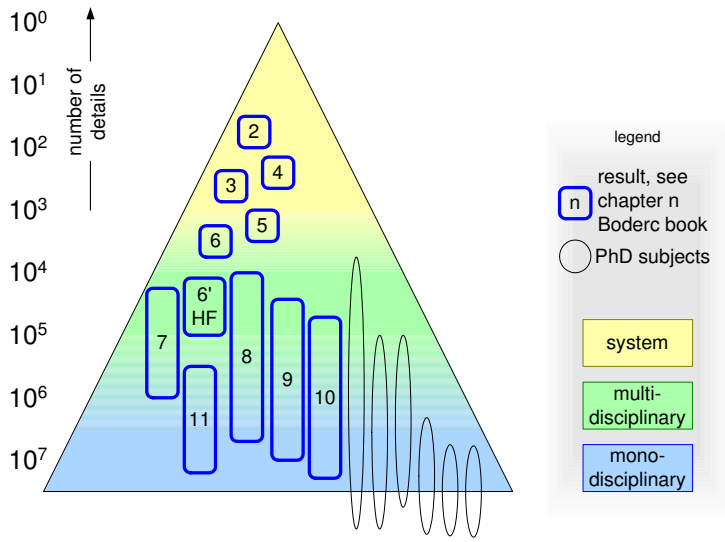


Figure 18.1: The project results positioned in the level of abstraction pyramid

The system-level reasoning used in the Boderc project was bundled in the Boderc method, that consists of a high-level framework, where more specific plug-ins are used to make it concrete and practical. The reasoning method as depicted graphically in Figure 2.7 was extracted in hindsight from the experience gained during the modeling activities. It collects more or less the general way of working that was observed in the project.

Submethods

Boderc explored a few submethods as system level plug-ins: the key drivers technique, threads-of-reasoning, and budget-based design (Chapters 3, 4 and 5, respectively). The key driver models for past as well as for future projects were highly appreciated by the industrial partner. The key driver method couples the main customer objectives to the technical requirements for the system and provides overview in the relationships between them. The submethod of threads-of-reasoning was used internally in the project, to relate industrial needs to (potential) research questions and modeling efforts. The value of these threads is the positioning of work and the relation between a local exploration and the more global context. Budget-based design was used mainly for power

considerations for a printer. However, we derived general guidelines (a method) on how to setup budgets and use them in a supportive manner for design purposes.

System-level models

The industrial appreciation of research results is a source of inspiration for further research, as can be seen by the results on kinematic modeling (Happy Flow) as discussed in Chapter 6. To learn from this successful industrial model, we identified the success factors of this particular model in Chapter 6. This should form a stepping stone to arrive at clear guidelines on how to set up effective models in an industrial context. In other domains with similar kinematic problems, like in mailing systems, there is already a strong interest in the particular model. The success of (the type of models as) the Happy Flow model created a demand for developing a similar type of models for thermo- and power-modeling (Chapter 7), which is an ongoing activity within Océ. Other 'system-level' models are also considered in Chapter 8 and 9. Chapter 8 focussed on how to evaluate the overall control architecture in terms of response times, CPU load, etc. Chapter 9 described models that are related to printing quality. New printer technologies were assessed via 'virtual' printer models with respect to their printing quality.

Detailed modeling

The study of stepper motors in Chapter 10 has a somewhat less system-level flavor as the before mentioned plug-ins. Océ Technologies had important reasons to replace the DC motors by stepper motors. For this purpose, Chapter 10 investigates the possibilities and impossibilities of stepper motors and aims at building a profound understanding of stepper motors that lead to practical design rules.

Also the work of the PhD students was stretched more to the multi-disciplinary design domain (see Figure 18.1) than in conventional research at universities.

- Chapter 12 provided an overview of techniques for state-of-the-art performance analysis for embedded real-time system architectures. Based on these experiences, an indication was given which method is used best under which circumstances to successfully support the decision making process for the architecture.

- Chapter 13 presented a model-driven design approach for real-time systems. This approach enables the analysis of real-time systems and allows automatic software code generation from the model that preserves the properties analyzed in the model.

- Chapter 14 takes a control engineering view on the controlled system and reduces the real-time software behavior to a model consisting of a varying time-delay. This chapter proposed analysis methods and techniques for the synthesis of controllers that are robust against these time-varying delays (i.e. jitter and latencies caused by computation and communication).

- For the control design of the drives of the paper transport system, Chapter 15 proposed a hierarchical control paradigm based on supervisory control is proposed. A systematic analysis and design procedure based on low-level controllers for the motors in combination with high-level sheet control was proposed.

- Chapter 16 described the design and application of event-driven control, which allows for a varying sample time in controllers. Event-driven control can have major benefits with respect to resource utilization like processor and communication load, while still maintaining a good control performance.

- In Chapter 17, a systematic design trajectory was proposed for the combination of real-time controllers and physical / mechanical processes. A design path was indicated in which stepwise the original (simulation) models of both plant and controller are replaced by their real implementations.

Chapter 11 discussed ways to simulate real-time embedded software together with its environment, being of a physical / mechanical nature. One approach, Software-in-the-Loop, is now used at Océ as a way to early test the functionality of paper path control software. This leads to faster feedback and design cycles and therefore better products.

The above indicates that several activities were carried out that connect more detailed knowledge (mono-disciplinary models) with multi-disciplinary design choices (system level models). This is indispensable for the design process as outlined in the overall Boderc method. One specific example ('schoolbook example') was already discussed in Chapter 2. In the above mentioned work, successful multi-disciplinary results were achieved, based on a more detailed understanding. Primary value of these activities is to enable the multi-disciplinary reasoning, without the need to cope continuously with all details.

18.3 Industry-as-Laboratory research approach

The intention of the industry-as-laboratory approach, as also discussed in Chapter 1, is twofold:

- to better connect academic research to industrial needs and to focus on results with industrial feasibility.

- to unfreeze industrial participants from contemporary constraints and to be perceptive for unconventional techniques.

The reward for this investment is that academic researchers obtained triggers for new, industrially relevant research directions and that industrial engineers were stimulated to try out multi-disciplinary models and design methods in actual development projects. Examples of the former are event-driven control design in Chapter 16 and the evaluation of embedded systems architecture in Chapter 12 to mention just two. Examples

of the latter include the tool coupling ideas as described in Chapter 11, the use of key drivers in Chapter 3, budget-based design in Chapter 5 and many others. Especially, the fact that researchers at Océ could work in copier development projects without having to contribute to the development project directly, was very beneficial. These researchers obtained the exploratory freedom to try new techniques and methods on actual industrial problems without the tight time-to-market constraints that the developers themselves are faced with.

The research created useful industrial *models* on one hand, and did benefit the advancement of multi-disciplinary *methods* on the other. As an example, the success of the Happy Flow model (Chapter 6) had a direct effect on reducing the effort and time needed to design the paper transport system and the print job scheduling. At the methodological level, Happy Flow was used to identify properties that effective industrial design models should satisfy. From these properties, guidelines can be derived on how to build successful industrial models. Also the making of actual budget models in the project (Chapter 5) was successful in itself, but resulted also in a more methodological view upon the use of budgets.

Along the lines of the industry-as-laboratory research approach, we will evaluate the original research hypothesis against the findings in the Boderc project. The research hypothesis of Boderc (see Chapter 1) was formulated as

The product creation lead time will be reduced significantly by the use of multi-disciplinary models during the early product development phases.

The question arises whether or not the research hypothesis is true and if it is true, what possible evidence is brought by the results of the Boderc project, as described in this book. A very strong 'true' can be given, even if we only focus on the one of the Boderc modeling activities: the Happy Flow model. Initial experience shows that significant savings in product creation lead time. On top of this reduction in the product creating lead time by Happy Flow, we believe that the use of the other Boderc models, like the virtual printer models, the heat flow modeling, the investigation in stepper motors, the evaluation of embedded system architectures, to mention a few, reduce the product creation time even further. Computations were not made to assert the economical value of these and other Boderc modeling activities. However, considering the broad use of the models within Océ, we conjecture that they must have a positive effect on the reduction of the product creating time, as otherwise developers and engineers would not have embraced them.

As the developed models predict the performance and consequences of specific design choices more accurately than previous state-of-the-practice models, uncertainty and risks are reduced for later stages. This means that less conservative designs become feasible resulting in better products. For instance, the Happy Flow enabled a better prediction of the paper transport systems and as such smaller printers could be built.

18.4 Lessons learned in process and organization

Of course, the development of model-based design methodologies for high-tech systems cannot be solved by one project like Boderc. Boderc made one proposal for a design methodology based on the experience obtained. Although a first step has been made, additional projects are needed to do *research* on methods. These additional projects must apply the researched methodology in different settings, and re-evaluate the hypothesis. The industry-as-laboratory approach has a long term character:

- Each industrial application requires significant time and effort to understand the necessary domain specific knowledge.
- Multiple industrial applications are required to support methodological conclusions.

For the benefit of future large-scale industrial research projects, we will collect our lessons learned in the Boderc project. This is especially important as Boderc is innovative in the process model that it uses for performing research.

Tension between mono-disciplinary academia and multi-disciplinary industry

The tension in this type of project is between the need for depth for mono-disciplinary academic partners and the need for short term industrially applicable and multi-disciplinary results of the industrial partners. The tension is most severe for students pursuing their PhD degree, as they are typically defending it within mono-disciplinary faculties. As a consequence, this tension is visible in the positioning of the subjects of the PhD-students, as shown in Figure 18.1. The required scientific depth pulls the students downward into the mono-disciplinary field. However, as can be observed there are some PhD results that stretches over several orders in the design pyramid. This is a clear benefit of a project like Boderc: the eye towards industrial applicability and system-level design is more profound in the Boderc (sub)projects than in the traditional research at universities. However, towards the end of the project the PhD students retracted more and more towards their own individual work on the PhD thesis, which is understandable on one hand, but caused disintegration of the project team on the other.

To value system-level research more at PhD level, an opportunity lies in creating the possibility of receiving a PhD degree in ‘multi-disciplinary or system engineering schools’ that go beyond the traditional engineering faculties as often encountered at universities.

Duration of the project

If we consider the development of the project members from mono-disciplinary towards multi-disciplinary, then we see that we needed at least two years for this growth. When we started we expected that this growth would take only one year. This means that we need more time for the total project than the 4 years as originally planned.

After two learning years at least two years of exploration and application are needed, followed again by at least one year of consolidation. A total project duration of 5 to 6 years would solve this problem, at least if we target for the original level of multi-disciplinary methods. However, this clashes with the need for short-term usable results as is often desirable from an industrial point of view.

Multi-disciplinary curriculum

Another solution to reduce the long learning phase could be the educational part of the PhD students. The first year was typically filled with mono-disciplinary classes within their own domain as this is customary for the PhD students in general. For future projects we recommend to create a multi-disciplinary curriculum for the PhD-students working in ESI projects. This would give the project members basic knowledge of other design disciplines. As a consequence, we expect that they (better) oversee consequences of design choices for other disciplines. Building a common multi-disciplinary device in the first year would also be a good means to learn cross-disciplinary thinking. The purpose of such a curriculum is twofold. First, a faster learning curve in the multi-disciplinary industrial setting and secondly, scientific results that fit higher in the design pyramid of Figure 18.1. Ideally we would like PhD students with a T-shaped thesis: sufficient depth in the mono-discipline, the vertical part of the T, connected to the multi-disciplinary problem, the horizontal ledger of the T.

Clear initial problem statement

Another remedy for the long learning phase is to have a clear problem statement at the beginning of a project. In the beginning of the Boderc project we started with mainly a collection of industrial problems that were faced during the final integration of a high-tech system, where the (sub)designs of the disciplines meet. We still had to extract the problem statement approach from these symptoms. We anticipated that the integration problems were caused by design decisions in the early design phases of which the consequences were not considered thoroughly across disciplines. From that we inferred the problem statement. Particularly in large-scale research projects, we recommend to prepare a sharp problem statement and approach before the project has even started.

Project team composition

Another point of discussion is the team composition. For instance, one could question if PhD students are the right persons of doing this type of research as they have the requirement to write a PhD thesis. From an academic point of view, PhD students are desirable as it forms one of the foundations of academic groups. But the requirement of developing sufficient novel contributions in a mono-disciplinary area forms an obstacle in obtaining system-level design techniques and models. A better balance could be obtained by involving more postdocs in the research as a remedy.

The members of the carrying industrial partner (CIP, the industrial partner that indicated the research problems) were typically young researchers, which had not yet developed their system engineering or system architecting skills extensively at the beginning of the project. Also they still had to explore the application field of printers and copiers. This resulted in the fact that domain specific knowledge was not readily available. More experienced engineers, instead of young researchers, is a solution although it is harder to unfreeze them from their project duties. Although it seems a high investment for industry to make their key engineers available for research projects, we believe that in the long run this would be very beneficial for all parties involved including themselves. Typically the first industrial Boderc workshop in which the CIP developers with more system overview were present, resulted in sharper discussion that arrived easier at the essence of the industrial design problem. A benefit for industry is that the young researchers were confronted with academic thinking, system level reasoning and industrial practice. These assets make them very valuable for the industry.

The non-CIP industrial people had more industrial and system-level experience. They turned out to be catalysts in the process of the project (especially in the beginning). The non-CIP industrial people are typically allocated to the project for two days per week. They found it hard to contribute in their part-time allocation. Part of the available time is needed for communication and recapturing what the other project members have been doing during their absence. The time left is not sufficient to actually build models. The project could benefit more from the existing industrial know-how if these industrial participants would also be full-time available. In hindsight we might have created a more balanced team in terms of experience by replacing one or two PhD students by post docs, but also getting more senior CIP people in the project (at more days per week).

Summary lessons learned

In summary, the lessons learned with respect to process and organization:

- Even more attention is needed for the composition of the project team, in the balance experience-inexperienced, in the balance industrial-academic and in the balance mono-disciplinary and multi-disciplinary.

- The industrial problem is rather broad and also the original project goal was not really crystallized at the start of the project. This hampered the fast start of the project. When the goal has to be discussed in the beginning of the project, it is better to let the PhD students start later.

- Also the research topics of PhD students should be clear at the start of the project and most importantly, should match the overall research goal. In the first year of Boderc the PhD topics were selected, while we believe that in year two we were better prepared to make the selection. As PhD students form a major part

of the work force and should take care of the momentum in the project, it is very important that they contribute directly to the overall project goal.

- The project team was too *dynamics and control engineering* oriented due to an unclear initial problem formulation. It is desirable to have more disciplines in the project team to arrive at a better balance.
- Part-time people can only be effective in a coaching role. The real research work (exploration, application, and consolidation) requires full-time people.
- Communication across disciplinary boundaries is really very difficult, as experienced throughout the project.
- It is very beneficial to provide plans for the classes that the PhD students attend in the first year. These classes should fit the overal multi-disciplinary project problem. In particular, some basic classes with respect to the specific application domain and classes outside the student's own discipline are considered valuable.
- The mix of project members in disciplines and background in the first year was a good preparation for the first industrial Boderc workshop. A critical success factor of this workshop was the presence of CIP engineers with system level overview. The participants were able to iterate between system requirements and mono-disciplinary design choices.

18.5 Boderc impact

Besides the research results described previously, Boderc made a broad impact and resulted in various spin-off activities. We will list here several examples of its impact and spin-off.

Academic impact:

- Six PhD students will defend their theses that include multi-disciplinary design knowledge
- Publications in leading journals in various research fields: embedded software engineering, control engineering, system engineering, etc. See Appendix A for an overview.
- Numerous papers and presentations at international conferences in the domains of both system architecting and more mono-disciplinary engineering disciplines like software engineering, control engineering, mechanical engineering and electrical engineering. See Appendix A.
- At the Forum on Specification and Design Languages (FSDL) 2006 a Boderc paper even won 'The Best Paper Award.'

- One of the Boderc partners started a course in the curriculum of the department of mechanical engineering of the TU/e on Embedded Motion Control to educate mechanical engineers in the implementation aspects of embedded controllers and the consequences of control algorithms on other design disciplines. Several Boderc members acted as lecturers for this course and in this way over 40 M.Sc. students were influenced by Boderc knowledge. This course will be continued in the future due to its success. Also at the universities of Nijmegen and Twente similar educational activities are started.
- Several postgraduate students in computer science (OOTI) from the Stan Ackermans Institute, numerous master and practical students from various academic groups participated and interacted in the Boderc project .
- We organized 3 successful symposia with on average 100 participants each.
- Boderc initiated the ViewCorrect [21] project, an STW-project between two academic groups participating in Boderc on tool coupling. Océ is a partner in this project.
- One of the research fellows stayed at Océ for 6 months to interchange ideas on system design. Sponsored by the Casimir program (dutch ministeries of economical affairs (EZ) and education, culture and sciences (OCW)).
- Two experimental set-ups of small paper paths were built at the universities of Eindhoven and Twente. Besides conducting research experiments, these are also used for educational purposes.

Industrial impact:

- The industrial participants in the project learned valuable system engineering skills, both at the CIP and the other participating companies.
- The financial return on investment was already obtained with one of the models, the Happy Flow model (Chapter 6).
- The success of the Happy Flow model resulted in the knowledge of the type of system behavior models that are useful for product development (see also Chapter 6). This awareness will lead to the development of more of this type of models. The energy model mentioned in Chapter 7 is one of them.
- The Software-in-the-Loop approach described shortly in Chapter 11 has been proven to be successful in a prototype and has been incorporated in the embedded software engineering process at Océ. It contributes to the development process by enabling much faster development cycles because of the rapid feedback it provides.
- Other Boderc modeling activities are being used or considered in Océ projects. An example is the evaluation of event-driven control (Chapter 16) in a prototype printer.

- Industrial workshops and 10 user group meetings were organized to transfer knowledge to Océ. This influences current and future design projects at Océ.
- By means of ESI courses, the acquired knowledge will be spread towards other industries in the Netherlands.

The industrial and academic outcomes of the project are such that follow-up research will take place that is based on a similar process model.

18.6 Concluding remarks

Of course, a pioneering project like Boderc is based on the 'spirit of an entrepreneur.' With good faith we started the project. Since there was little to no experience with research projects of this size and type, the Boderc project had a high learning character. As such, it is inevitable that there is room for improvement in the process and organizational part. Most importantly, we have to learn from this experience for the future.

The next generation of projects of the Embedded Systems Institute already benefits from the lessons learned in Boderc, which underlines the innovative nature of the Boderc project. Although the Boderc project may have suffered a bit from its pioneering position, we can be very satisfied with the outcomes, as described in the previous sections of this book.

If one takes a 'business-oriented view' on the Boderc project, one can say that it generated return on investment for the involved companies and academic groups that are clearly above expectation. For the academic groups this was typically realized via graduations of master and PhD students, published papers and the Boderc impact on their curricula. ESI was able to attract 3 research fellows for its staff via the Boderc project and Boderc helped to put ESI on the world map as a leading center in the area of embedded system engineering. Océ saved a lot of design effort and time in current and future projects due to the development of many valuable models, techniques and methods.

Using a more 'soft view,' the Boderc project created a lot of *awareness* both within academia and industry with mutual understanding and respect for individual positions, capabilities and strengths. The difficulties in multi-disciplinary and system-level design became more explicit and as such created a first step in addressing them. Academia were confronted with industrial needs, while industry learned to untangle itself occasionally from the time pressure present in product development projects. This lead to the development of models, techniques and methods that were truly relevant in industrial practice. It also initiated the cooperation of all disciplines, already from the earliest phases of design projects. Of course, the derived methodology, understanding and models should be refined further and validated in future projects. But all aspects taken into account, the Boderc project made an excellent first step in developing model-based methodologies for designing high-tech systems.

Appendix A

List of Boderc publications

The Boderc publications are categorized into the specific research fields they belong to. This illustrates the impact of Boderc on various research disciplines. Within the categories *system engineering*, *control engineering* and *hardware-software engineering*, we ordered the publications anti-chronologically.

System Engineering

1. Muller G.J., Heemels, W.P.M.H.
   Five Years of Multi-Disciplinary Academic and Industrial Research: Lessons Learned. Conference on System Engineering Research (CSER) 2007.

2. Verhoef, M., Larsen, P.G.
   Interpreting Distributed System Architectures with VDM++- A case study. Conference on System Engineering Research (CSER) 2007.

3. Bosch, P.F.A. van den, Verhoef, M., Muller G.J., Florescu, O.
   Modeling of hardware software performance of high-tech systems. Seventeenth symposium International Council on System Engineering (INCOSE) 2007, San Diego, USA.

4. Heemels, W.P.M.H., Somers, L., Bosch, P.F.A. van den, Muller, G., Yuan, Z., Wijst, B. van der, and Brand, A. van den.
   The use of the key driver technique in the design of copiers. In Proceedings of the International Conference on Software and Systems Engineering and their Applications (ICSSEA), December 5-7, 2006, Paris, France.

5. Freriks, H., Heemels, W.P.M.H and Muller, G.J., Sandee,J.H.
   On the Systematic Use of Budget-Based Design. Sixteenth symposium International Council on System Engineering (INCOSE) 2006, Orlando, USA.

6. Sandee, J.H., Heemels, W., Muller, G., Bosch, P.F.A. van den, and Verhoef, M. *Threads of reasoning: A case study in printer control*. Sixteenth symposium International Council on System Engineering (INCOSE) 2006, Orlando, USA.

7. Heemels W.P.M.H., Waal E. van de, and Muller, G.J. *A multi-disciplinary and model-based design methodology for high-tech systems*. In Proceedings of the Conference on System Engineering Research (CSER), April 2006, Los Angeles, California.

8. Bosch, P.F.A. van den and Waal, E.H. van de. *A case study of multi-disciplinary modeling using MATLAB / Simulink and TrueTime*. Fifteenth symposium International International Council on System Engineering (INCOSE) 2005, Rochester, USA.

9. Muller G.J. *Do useful Multi-Domain Methods exist?* In Proceedings of the Conference on System Engineering Research (CSER), March 2005, Hoboken, USA.

Control engineering

1. Visser, P. M. and Broenink, J.F. *Controller and Plant System Design Trajectory* In Proc. IEEE International symposium on Computer Aided Control Systems Conference, CACSD 2006.

2. Cloosterman, M.B.G., Wouw, N. van de, Heemels, W.P.M.H. and Nijmeijer, H. *Robust Stability of Networked Control Systems with Time-varying Network-induced Delays*. In Proceedings of the 45th IEEE Conference on Decision and Control (CDC), 2006, San Diego, U.S.A.

3. Sandee, J.H., Visser, P.M. and Heemels, W.P.M.H. *Analysis and experimental validation of processor load for event-driven controllers*. Proceedings of the IEEE Conference on Control and Applications 2006, Munich, Germany, pages 1879-1884.

4. Heemels, W.P.M.H. and Sandee, J.H. *Practical stability of perturbed event-driven controlled linear systems*. Proceedings of the American Control Conference 2006 in Minneapolis, USA, pages 4379-4386.

5. Bukkems, B.H.M, van de Molengraft, M.J.G., Heemels, W.P.M.H., Wouw, N. van de, and Steinbuch, M. *A Piecewise Linear Approach towards Sheet Control in a Printer Paper Path*. Proceedings of the American Control Conference 2006 in Minneapolis, USA, pages 1315-1320.

6. Bukkems, B.H.M, de Best, J.J.T.H., van de Molengraft, M.J.G., and Steinbuch,

M.
Robust Piecewise Linear Sheet Control in a Printer Paper Path. Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems 2006 in Alghero, Sardinia, Italy, pages 142-147.

7. Sandee, J.H., Heemels W.P.M.H., Bosch, P.P.J. van den.
Event-driven control as an opportunity in the multidisciplinary development of embedded controllers. Proceedings of the American Control Conference 2005, vol. 3, Portland, 2005, pages1776-1781.

8. Bukkems, B., Sandee, J.H., Beckers, J., Yuan, Z., Wijst, B. van der.
Multi-disciplinary modelling of dynamic embedded systems. Proceedings of Mechatronics and Robotics 2004, Aachen, Germany, pages 27, part I.

Hardware-software engineering

1. Verhoef, M., Larsen, P.G., Hooman, J.
Modeling and validating distributed embedded real-time systems with VDM++. In Misra, J., Nipkow, T., Sekerinski, E., editors, Formal Methods (FM) 2006, volume 4085 of Lecture Notes in Computer Science (LNCS), pages 147–162. Springer, 2006. This paper is available on-line at `http://dx.doi.org/10.1007/11813040_11`.

2. Florescu, O., Voeten, J.P.M., Verhoef, M., Corporaal, H.
Reusing real-time systems design experience through modelling patterns. Proceedings of the Forum on Specification & Design Languages 2006 (FDL'06), ISBN 3-00-019710-9, pages 375-380, Darmstadt, Germany, September 2006.

3. Florescu, O., Voeten, J.P.M., Corporaal, H.
Property-preservation synthesis for unified control- and data-oriented models. Editor Vachoux, Alain, *Applications of specification and design languages for SoCs*, ISBN 1-4020-4997-8, pages 247-262, Springer, 2006.

4. Florescu, O., Huang, J., Voeten, J.P.M., Corporaal, H.
Strengthening property preservation in concurrent real-time systems. Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), ISBN 0-7695-2676-4, pages 106-109, Sydney, Australia, August 2006.

5. Florescu, O., Hoon, M. de, Voeten, J.P.M., Corporaal, H.
Probabilistic modelling and evaluation of soft real-time embedded systems. Proceedings of the Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS VI), LNCS 4017, ISBN 3-540-36410-2, pages 206-215, Samos, Greece, July 2006.

6. Florescu, O., Hoon, M. de, Voeten, J.P.M., Corporaal, H.

Performance modelling and analysis using POOSL for an in-car navigation system. Proceedings of the 12th Annual Conference of the Advanced School for Computing and Imaging (ASCI), ISBN 90-810849-1-7, pages 37-45, Lommel, Belgium, June 2006.

7. Hendriks, M, Verhoef, M.
Timed automata based analysis of embedded systems architectures. In Workshop of Parallel and Distributed Real-Time Systems (WPDRTS). IEEE, 2006. Online proceedings to appear. Technical report ICIS-R06003 is available on-line at `http://www.cs.ru.nl`.

8. Verhoef, M.
On the use of VDM++ for specifying real-time systems. In John Fitzgerald, Peter Gorm Larsen and Nico Plat, editors, Towards Next Generation Tools for VDM: Contributions to the First International Overture Workshop, CS-TR 969, pages 26–43. School of Computing Science, Newcastle University, June 2006. This technical report is available on-line at `http://www.cs.ncl.ac.uk/research/pubs/`.

9. Wandeler, E., Thiele, L., Verhoef, M., and Lieverse, P.
System architecture evaluation using modular performance analysis: a case study. International Journal of Software Tools for Technology Transfer (STTT), 8(6):649–667, November 2006. This paper is available on-line at `http://dx.doi.org/10.1007/s10009-006-0019-5`.

10. Yang Huang, Visser, P. M. and Broenink, J.F.
A Clock Synchronization Skeleton Based on RTAI. In 8th Real-Time Linux Workshop.

11. Florescu, O., Voeten, J.P.M., Corporaal, H.
Property-preservation synthesis for unified control- and data-oriented models. Proceedings of the Forum on Specification & Design Languages 2005 (FDL'05), ISSN 1636-9874, Lausanne, Switzerland, September 2005.

12. Huang, J., Voeten, J.P.M., Florescu, O. and Putten, P.H.A. van der, Corporaal, H.
Predictability in real-time system development. Editor Boulet, Pierre, Advances in design and specification languages for SoCs, ISBN 0-387-26149-4, Springer, 2005.

13. Yuchen, Z., Orlic, B., Visser, P.M. and Broenink, J.F.
Hard Real-Time Networking on Firewire. In 7th Real-Time Linux Workshop, 2005

14. Florescu, O., Voeten, J.P.M., Corporaal, H.
A unified model for analysis of real-time properties. Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods (ISoLa

2004), TR-2004-6, pages 220-227, Paphos, Cyprus, November 2004.

15. Florescu, O., Voeten, J.P.M., Huang, J., Corporaal, H.
Error estimation in model-driven development for real-time software. Proceedings of the Forum on Specification & Design Languages 2004 (FDL'04), ISSN 1636-9874, Lille, France, September 2004.

16. Visser, P. M., Groothuis, M. A. and Broenink, J.F.
FPGAs as versatile configurable I/O devices in Hardware-in-the-Loop Simulation. In The 25th IEEE International Real-Time Systems Symposium, 2004

17. Visser, P. M., Groothuis, M. A. and Broenink, J.F.
Multi-Disciplinary Design Support using Hardware-in-the-Loop Simulation. In 5th PROGRESS Symposium on Embedded Systems, 2004

18. Hooman, J., Mulyar, N., and Posta, L.
Supporting model-based simulation of embedded systems by coupling tools. In Proceedings of the 5th PROGRESS Symposium on Embedded Systems, pages 131–134. Technology Foundation STW, 2004.

19. Hooman, J., Mulyar, N., and Posta, L.
Validating UML models of embedded systems by coupling tools. In Proceedings Workshop on Specification and Validation of UML models for Real-Time and Embedded Systems (SVERTS). Verimag, 2004.

20. Hooman, J., Mulyar, N., and Posta, L.
Coupling Simulink and UML models. In B. Schnieder and G. Tarnai, editors, Proceedings of Symposium FORMS/FORMATS 2004, pages 304–311, 2004.

Appendix B

List of authors

Prof. dr. ir. J. van Amerongen
University of Twente
j.vanamerongen@utwente.nl

Ing. J.M.J. Beckers
Océ Technologies B.V.
jan.mj.beckers@oce.com

Ir. J.J.T.H. de Best
Eindhoven University of Technology
j.debest@tue.nl

Prof. dr. ir. P.P.J. van den Bosch
Eindhoven University of Technology
p.p.j.v.d.bosch@tue.nl

Ir. P.F.A. van den Bosch
Océ Technologies B.V.
peter.vandenbosch@oce.com

Ir. A. van den Brand
Centric Tsolve
adriaan.van.den.brand@TSolve.com

Dr. ir. J.F. Broenink
University of Twente
j.f.broenink@utwente.nl

Ir. B.H.M. Bukkems
Eindhoven University of Technology
b.h.m.bukkems@tue.nl

Ir. M.B.G. Cloosterman
Eindhoven University of Technology
m.b.g.cloosterman@tue.nl

Prof. dr. H. Corporaal
Eindhoven University of Technology
h.corporaal@tue.nl

Ir. O. Florescu
Eindhoven University of Technology
o.florescu@tue.nl

Ir. H.J.M. Freriks
Océ Technologies B.V.
hennie.freriks@oce.com

Dr. ir. W.P.M.H. Heemels
Eindhoven University of Technology
Embedded Systems Institute
m.heemels@tue.nl

Dr. J.J.M. Hooman
Radboud University Nijmegen
Embedded Systems Institute
jozef.hooman@esi.nl

Ir. K.J. Klein Koerkamp
Océ Technologies B.V.
koen.kleinkoerkamp@oce.com

Dr. ir. M.J.G. van de Molengraft
Eindhoven University of Technology
m.j.g.v.d.molengraft@tue.nl

Dr. G.J. Muller
Embedded Systems Institute
gerrit.muller@esi.nl

Prof. dr. H. Nijmeijer
Eindhoven University of Technology
h.nijmeijer@tue.nl

Ir. J.H. Sandee
Eindhoven University of Technology
j.h.sandee@tue.nl

Dr. ir. L.J.A.M. Somers
Océ Technologies B.V.
lou.somers@oce.com

Prof. dr. ir. M. Steinbuch
Eindhoven University of Technology
m.steinbuch@tue.nl

J. Stolte
Eindhoven University of Technology
jstolte@tue.nl

Dr. ir. A. Veltman
Eindhoven University of Technology
a.veltman@tue.nl

Ir. M.H.G. Verhoef
Chess
marcel.verhoef@chess.nl

Dr. ir. J.P.M. Voeten
Eindhoven University of Technology
j.p.m.voeten@tue.nl

Ir. P.M. Visser
University of Twente
p.m.visser@utwente.nl

Ir. E.H. van de Waal
Imtech
evert.vandewaal@imtech.nl

Ir. B. van der Wijst
Philips Applied Technologies
berry.van.der.wijst@philips.com

Dr. ir. N. van de Wouw
Eindhoven University of Technology
n.v.d.wouw@tue.nl

Dr. Z. Yuan
Océ Technologies B.V.
zhaorui.yuan@oce.com

Boderc partners (institutions and companies):

Chess	Best, The Netherlands
Eindhoven University of Technology	Eindhoven, The Netherlands
Embedded Systems Institute	Eindhoven, The Netherlands
Imtech	The Hague/Baarn, The Netherlands
Océ Technologies B.V.	Venlo, The Netherlands
Radboud University Nijmegen	Nijmegen, The Netherlands
University of Twente	Enschede, The Netherlands

For more information: office@esi.nl

Bibliography

[1] P.P. Acarnley and P. Gibbons. Closed-loop control of stepping motors: prediction and realisation of optimum switching angle. *IEEE Proceedings*, Part B, 129(4), 1982.

[2] I. Alexander. Towards automatic traceability in industrial practice. *Proceedings of the First International Workshop on Traceability, Edinburgh*, pages 26–31, 2002.

[3] G. Altshuller. The innovation algorithm. triz, systematic innovation and technical creativity. 2000.

[4] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] D. Amyot and G. Mussbacher. Urn: Towards a new standard for the visual description of requirements. *LNCS 2599*, pages 21–37, June 2002.

[6] A. Antón, J. Dempster, and D. Siege. Managing use cases during goal driven requirements engineering: Challenges encountered and lessons learned. *TR-99-16, North Carolina State Univ*, December 1999.

[7] Karl J. Åström and Björn Wittenmark. *Computer Controller Systems: Theory and Design*. Prentice Hall, Upper Saddle River, New Jersey, 3rd edition, 1997.

[8] S. Baruah, D. Chen, and A. Mok. Jitter concerns in periodic task systems. *Proceedings of the Eighteenth Real-Time Systems Symposium, San Francisco, CA*, pages 68–77, 1997.

[9] I. Bate and N. Audsley. Flexible design of complex high-integrity systems using trade offs. *Proc. 8th Int. Symp. High Assurance Systems Engineering*, 2004.

[10] J.O. Bayer, P. Flege, R. Knauber, D. Laqua, K. Muthig, T. Schmid, Widen, and J.M. DeBaud. Pulse: A methodology to develop software product lines. *Proceedings of the symposium on software reusability*, pages 122–131, 1999.

[11] G. Behrmann, A. David, and K.G. Larsen. A Tutorial on UPPAAL. In *Formal Methods for the Design of Real-time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[12] Matthijs H. ten Berge, Bojan Orlic, and Jan F. Broenink. Co-simulation of networked embedded control systems, a CSP-like process-oriented approach. In *Proc. of joint CCA, CACSD and ISIC*, pages 434–439, Munich, Germany, October 2006.

[13] S.S. Blackman. *Multiple target tracking with radar applications*. Norwood, MA: Artech House, 1986.

[14] H. Bode. *Network Analysis and Feedback Amplifier Design*. Van Nostrand Reinhold, New York, 1945.

[15] Leo J. van Bokhoven, Jeroen P.M. Voeten, and Marc C.W. Geilen. Software synthesis for system level design using process execution trees. In *Proc. of 25th Euromicro Conference*, pages 463–467, 1999.

[16] L. Boltzmann. Ableitung des stefanschen gesetzes betreffend die abhängigkeit der wärmestrahlung. *Annalen der Physik und Chemie*, 22, 1884.

[17] Egor Bondarev, Michel Chaudron, and Peter de With. Quality-oriented design space exploration for component-based architectures. Technical report, Technical University of Eindhoven, Department of Mathematics and Computer Science, February 2006.

[18] P.F.A. van den Bosch and E.H. van de Waal. A case study of multi-disciplinary modelling using matlab/simulink and truetime. *Proc. INCOSE Symposium 2005*, 2005.

[19] P.P.J. van den Bosch and A.C. van der Klauw. *Modeling, Identification and Simulation of Dynamical Systems*. CRC Press Inc., 1994.

[20] P.C. Breedveld. *Dynamische systemen : modelvorming en simulatie met bondgrafen*. Open universiteit, The Netherlands, 1994.

[21] J.F. Broenink and J.P.M. Voeten. Viewcorrect: Predictable co-design for distributed embedded control systems.

[22] D. de Bruin and P.P.J. van den Bosch. Measurement of the lateral vehicle position with permanent magnets. In *Proceedings of IFAC workshop on Intelligent Components for Vehicles*, pages 9–14, Seville, Spain, 1998.

[23] R.J.A. Buhr. Use case maps as architectural entities for complex systems. *IEEE Transactions on Software Engineering*, 24, Issue 12:1131 – 1155, December 1998.

[24] B.H.M. Bukkems, M.J.G. van de Molengraft, W.P.M.H. Heemels, N. van de Wouw, and M. Steinbuch. A piecewise linear approach towards sheet control in a printer paper path. In *Proc. of the American Control Conference*, pages 1315–1320, Minneapolis, USA, 2006.

[25] B.H.M. Bukkems, J.H. Sandee, J.B.C. Beckers, Z. Yuan, B. van der Wijst, and M.J.G. van de Molengraft. A case-study in multidisciplinary modeling of dynamic embedded systems. *IEEE Conference on Mechatronics and Robotics 2004, Aachen, Germany.*, 2004.

[26] Björn Bukkems. *Sheet Feedback Control Design in a Printer Paper Path*. (To be published), Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2007.

[27] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[28] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzen. How does control timing affect performance? *IEEE Control Systems Magazine*, pages 16–30, June 2003.

[29] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. 6th Design, Automation and Test in Europe*, pages 190–195, 2003.

[30] Carlo Cloet. *A Mechatronics Approach to Copier Paperpath Design*. PhD thesis, University of California, Berkeley, CA, USA, 2001.

[31] Marieke Cloosterman, Nathan van de Wouw, Maurice Heemels, and Henk Nijmeijer. Robust control of networked control systems with uncertain time-varying delays. *DCT internal report 2006-121*, 2006.

[32] Marieke Cloosterman, Nathan van de Wouw, Maurice Heemels, and Henk Nijmeijer. Robust stability of networked control systems with time-varying network-induced delays. In *Proc. of the 45th Conference on Decision and Control*, San Diego, California, USA, December 2006.

[33] A. Cockburn. *Writing effective use cases.* Addison-Wesley, 2000.

[34] P. Crnosija. Microcomputer implementation of optimal algorithms for closed-loop control of hybrid stepper motor drives. *IEEE Transactions on Industrial Electronics*, 47(6), 2000.

[35] J. Dahmann, R. Fujimoto, and R. Weatherly. The department of defense high level architecture. In *The 1997 Winter Simulation Conference*, pages 142–149, 1997.

[36] R.L. Dillon, M.E. Paté-Cornell, and S.D. Guikema. Programmatic risk analysis for critical engineering systems under tight resource constraints. *Operations Research*, 51, No. 3:354–370, 2003.

[37] R.C. Doff, M.C. Fatten, and C.A. Phillips. Adaptive sampling frequency for sampled-data control systems. *IRE Transactions on Automatic Control*, AC-7:38–47, 1962.

[38] P.G. Engeldrum. *Psychometric scaling: A Toolkit for Imaging Systems*. Imcotek Press, 2000.

[39] Oana Florescu, Menno de Hoon, Jeroen Voeten, and Henk Corporaal. Performance modelling and analysis using poosl for an in-car navigation system. In *Proceedings of the 12th Annual Conference of the Advanced School for Computing and Imaging (ASCI)*, pages 37–45, June 2006.

[40] Oana Florescu, Menno de Hoon, Jeroen Voeten, and Henk Corporaal. Probabilistic modelling and evaluation of soft real-time embedded systems. In *Proceedings of the Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS VI)*, LNCS 4017, pages 206–215, July 2006.

[41] Oana Florescu, Jinfeng Huang, Jeroen Voeten, and Henk Corporaal. Strengthening property preservation in concurrent real-time systems. In *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 106–109, August 2006.

[42] Oana Florescu, Jeroen Voeten, and Henk Corporaal. Modelling patterns for analysis and design of real-time systems. Technical Report ESR-2006-05, Eindhoven University of Technology, 2006.

[43] Oana Florescu, Jeroen Voeten, Jinfeng Huang, and Henk Corporaal. Error estimation in model-driven development for real-time software. In *Proceedings of the Forum on Specification & Design Languages 2004 (FDL'04)*, September 2004.

[44] Oana Florescu, Jeroen Voeten, Marcel Verhoef, and Henk Corporaal. Reusing real-time systems design experience through modelling patterns. In *Proceedings of the Forum on Specification & Design Languages 2006 (FDL'06)*, September 2006.

[45] International Organisation for Standardization. Space systems - mass properties control. Proposal ISO International Standard, ISO/CD 22010.

[46] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Prentice Hall, Upper Saddle River, New Jersey, USA, 2002.

[47] H.J.M. Freriks. White paper on designing with stepper motors. Technical report, 2005.

[48] H.J.M. Freriks, W.P.M.H. Heemels, and G.J. Muller. On the systematic use of budget-based design. *Proceedings of 16th annual international symposium of the INCOSE*, 2005.

[49] B. Friedland. Optimum steady-state position and velocity estimation using sampled position data. *IEEE transactions on Aerospace and Electronic Systems*, AES-9(6):906–911, 1973.

[50] P. Gahinet, A. Nemirovski, A. J. Laub, and M. Chilali. *LMI Control Toolbox for Use with Matlab*. The Mathworks Inc., Natick, MA, USA, May 1995.

[51] Marc G.W. Geilen. *Formal Techniques for Verification of Complex Real-Time Systems*. PhD thesis, Eindhoven University of Technology, Eindhoven NL, 2002.

[52] Geilen, M.C.W. and Voeten, J.P.M. and Putten, P.H.A. van der and Bokhoven, L.J. van and Stevens, M.P.J. Poosl. *Computer Languages 27*, pages 19–38, 2001. http://www.es.ele.tue.nl/poosl.

[53] T. Glad and L. Ljung. Velocity estimation from irregular, noisy position measurements. In *Proceedings of the IFAC 9th Triennial World Congress*, pages 1069–1073, Budapest, 1984.

[54] Yoram Halevi and Asok Ray. Integrated communication and control systems: Part i and ii. *Journal of Dynamic Systems, Measurement, and Control*, 110(4):367–381, 1988.

[55] W.P.M.H. Heemels, R.J.A. Gorter, A. van Zijl, P.P.J. van den Bosch, S. Weiland, W.H.A. Hendrix, and M.R. Vonder. Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, 7:1467–1482, 1999.

[56] W.P.M.H. Heemels, E. v.d. Waal, and G.J. Muller. A multi-disciplinary and model-based design methodology for high-tech systems. *Proceedings of CSER*, 2006.

[57] Martijn Hendriks and Marcel Verhoef. Proc. timed automata based analysis of embedded system architectures. In *Workshop on Parallel and Distributed Real-Time Systems*, 2006.

[58] João P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. Networked control systems: Analysis and design. To appear in the *Proc. of IEEE*, Special Issue on Networked Control Systems, 2006.

[59] S. van der Hoest. The development of a software-in-the-loop simulation framework for testing real-time control software. Technical report, Stan Ackermans Institute, Eindhoven, 2006.

[60] J. Hooman, N. Mulyar, and L. Posta. Coupling Simulink and UML models. In B. Schnieder and G. Tarnai, editors, *Proceedings of Symposium FORMS/FORMATS 2004*, pages 304–311, 2004.

[61] Jozef Hooman, Hillel Kugler, Iulian Ober, Anjelika Votintseva, and Yuri Yushtein. Supporting uml-based development of embedded systems by formal techniques. *Software and Systems Modeling*, to appear.

[62] Jinfeng Huang, Jeroen P.M. Voeten, and Marc C.W. Geilen. Real-time property preservation in approximations of timed systems. In *Proc. of 1st Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 163–171, June 2003.

[63] A. Hughes and P.J. Lawrenson. Simple theoretical stability criteria for 1.8° hybrid motors. *Int. Symp. on Stepping Motors and Systems*, 1979.

[64] IBM/Rational. Rose realtime. http://www.ibm.com/.

[65] M. Jazayeri, A. Ran, and F. v.d. Linden. Software architecture for product families. 2000.

[66] C. Jongeneel. Klokloze chips. *De Ingenieur*, 117(4):52–53, 2005.

[67] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics, modeling and simulation of Mechatronic Systems*. John Wiley & sons, New York, third edition edition, 2000.

[68] G.A. Katopis and et al. MCM technology and design for the S/390 G5 System. *IBM Journal of Research and Development*, 43, No 5/6, September/November 1999.

[69] Bart Kienhuis, Ed Deprettere, Kees Vissers, and Pieter van der Wolf. An approach for quantitative analysis of application-specific dataflow architectures. In *Proceedings of the IEEE ASAP*, pages 338–349, 1997. International Conference on Application-Specific Systems, Architectures and Processors.

[70] T. Kostelijk. Misleading architecting tradeoffs. *IEEE Computer*, May 2005.

[71] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[72] M. Krucinski, C. Cloet, M. Tomizuka, and R. Horowitz. Asynchronous observer for a copier paper path. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 3, pages 2611–2612, Tampa, Florida, 1998.

[73] Martin Krucínski. *Feedback Control of Photocopying Machinery*. PhD thesis, University of California, Berkeley, CA, USA, 2000.

[74] C. van Lamsweerde. Goal oriented requirements engineering: A guided tour. *Proc. 5th IEEE Int. Symp. on Requirements Eng.*, pages 249–263, August 2001.

[75] Feng-Li Lian. *Analysis, Design, Modelingadn Control of Networked Control Systems*. PhD thesis, University of Michigan, Ann Arbor, USA, 2001.

[76] J.A. López-Orozco, J.M. de la Cruz, E. Besada, and P. Ruipérez. An asynchronous, robust, and distributed multisensor fusion system for mobile robots. *The international Journal of Robotics Research*, 19(10):914–932, 2000.

[77] M.W. Maier and E. Rechtin. *The Art of Systems Architecting*. CRC Press, Boca Raton, second edition, 2002.

[78] R. Malan and D. Bredemeyer. Software architecture action guide. 2005.

[79] E. Mohammed and et al. Optical interconnect system integration for ultra-short-reach applications. *Intel Technical J. on Optical Technologies and Applications*, 8, No 2, May 2004.

[80] René van de Molengraft, Bram de Kraker, and Maarten Steinbuch. Integrating experimentation into control courses. *IEEE Control Syst. Mag.*, 25(1):40–44, February 2005.

[81] G.J. Muller. *CAFCR: A multi-view method for embedded systems architecting; balancing genericity and specificity.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 2004.

[82] G.J. Muller. Do useful multi-domain methods exist? *Conference System Engineering Research (CSER), Hobroken, USA*, March 2005.

[83] G.J. Muller. Industry and academia: Why practitioners and researchers are disconnected. *Proc. INCOSE Symposium, Rochester, NY, USA.*, 2005.

[84] Payam Naghshtabrizi and Joao P. Hespanha. Designing an observer-based controller for a network control system. In *Proc. of the 44th Conference on Decision and Control, and the European Control Conference 2005*, pages 848–853, Seville, Spain, December 2005.

[85] Xavier Nicollin and Joseph Sifakis. An overview and synthesis on timed process algebras. In *Proc. of the Real-Time: Theory in Practice, REX Workshop*, pages 526–548, London, UK, 1992. Springer-Verlag.

[86] Johan Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

[87] OMG. *Unified Modeling Language (UML) - Version 1.5*. OMG document formal/2003-03-01, Needham MA, 2003.

[88] OMG. *UML Profile for Schedulability, Performance, and Time Specification - Version 1.1*. OMG document formal/2005-01-02, 2005.

[89] International Council on Systems Engineering (INCOSE) Technical board. *System Engineering Handbook*. INCOSE, 2004. A "what to" guide for all se practitioners.

[90] G.J.P.M. van Oosterhout. Spectral color prediction by advanced physical modeling of toner, ink and paper, with application to halftoned prints. *Proc. ISandT's NIP19*, pages 797–, 2003.

[91] David L. Parnas. Software engineering: An unconsummated marriage. *Communications of the ACM*, page 128, September 1997. This article can also be found in Software Fundamentals,Papers by David Parnas, Addison-Wesley.

[92] R. Phaal, C.J.P. Farrukh, and D.R. Probert. Technology roadmapping - a planning framework for evolution and revolution. *Technological Forecasting & Social Change*, 71, No 1-2:5–26, 2004.

[93] A.M. Phillips and M. Tomizuka. Multirate estimation and control under time-varying data sampling with applications to information storage devices. In *Proceedings of the 1995 American control conference*, volume 6, pages 4151–4155, 1995.

[94] Colin Potts. Software-engingeering research revisited. *IEEE Software*, Vol. 10, No. 5:19–28, September/October 1993.

[95] P.H.A. van der Putten and J.P.M. Voeten. Specification of reactive hardware/software systems - the method software/hardware engineering. 1997.

[96] QFD Institute. QFD institute. `http://www.qfdi.org/`, 2000.

[97] Sudhendu Rai and Warren B. Jackson. A hybrid hierarchical control architecture for paper transport systems. In *Proc. of the* 37th *IEEE Conference on Decision and Control*, pages 4249–4250, Tampa, Florida, USA, December 1998.

[98] J.B. ReVelle. *The QFD handbook*. Wiley, 1998.

[99] H. Saiedian, P. Kumarakulasingam, and M. Anan. Scenario-based requirements analysis techniques for real-time software systems: a comparative evaluation. *Requirements Eng. J.*, 10:22–33, 2005.

[100] J.H. Sandee. *Event-driven control in theory and practice - trade-offs in software and control performance*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.

[101] J.H. Sandee, W.P.M.H. Heemels, G.J. Muller, P.F.A. van den Bosch, and M.H.G. Verhoef. Threads of reasoning: A case study. *Proceedings of the 16th annual international symposium of INCOSE*, 2006.

[102] R. Sanz and K.-E. Årzén. Trends in software and control. *IEEE Control Systems Magazine*, 23(3):12–15, 2003.

[103] A.J. van der Schaft and J.M. Schumacher. An introduction to hybrid dynamical systems. *Lecture Notes in Control and Information Sciences*, page Vol. 251, 1999.

[104] S.A. Schweid and et al. Hybrid step motor position estimation from back emf. *IEEE Conf. on Control Applications*, 1995.

[105] B. Selic, G. Gullekson, and P.T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, 1994.

[106] Simulink. The Mathworks. http://www.mathworks.com/products/simulink/.

[107] S.T. Stanton and et al. Overlay error budgets for a high-throughput scalpel system. *Proceedings of SPIE*, 3676:543–555, June 1999.

[108] George Stephanopoulos. *Chemical Process Control*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1984.

[109] J. Stolte. Understanding instability in hybrid stepper motors. Technical report, Eindhoven University of Technology, 2006.

[110] Telelogic. Combining Rhapsody and Simulink. http://www.telelogic.com/.

[111] Lothar Thiele, Samarjit Chakraborty, and Martin Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems*, volume 4, pages 101–104, 2000.

[112] Yodyium Tipsuwan and Mo-Yuen Chow. Control methodologies in networked control systems. *Control Engineering Practice*, 11:1099–1111, 2003.

[113] TrueTime. http://www.control.lth.se/truetime/.

[114] UCM-URN. Use case maps. http://www.usecasemaps.org/index.shtml.

[115] Toronto University. Goal oriented requirements language.

[116] A. Veltman and P.P.J. van den Bosch. A universal method for modelling electrical machines. *Conf. on Electrical Machines and Drives*, 1992.

[117] Marcel Verhoef, Peter Gorm Larsen, and Jozef Hooman. Modeling and validating distributed embedded real-time systems with VDM++. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *Proc. Formal Methods 2006*, volume 4085 of *LNCS*, pages 147–162. Formal Methods Europe, Springer, 2006.

[118] Peter M. Visser and Jan F. Broenink. Controller and plant system design trajectory. In *Proc. IEEE Int'l Symposium on Computer Aided Control Systems Conference, CACSD 2006*, pages 1910–1915, Munich, 2006. IEEE.

[119] C.V. Vottis. *Extracting more accurate position and velocity information from optical incremental encoders*. SAI/2yr Thesis, Technische Universiteit Eindhoven, Netherlands, 2003.

[120] G.C. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. In *Proc. of the American Control Conference*, pages 2876–2880, San Diego, California, USA, June 1999.

[121] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. *Intern. journal on software tools technology transfer*, 2006.

[122] P.M. Wecksten, J. Vasell, and M. Jonsson. Towards a tool for derivation of implementation constraints. *Proc. 9th IEEE Int. Conf. Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age*, 2004.

[123] R. Wieringa. Requirements engineering: problem analysis and solution specification. *ICWE*, pages 13–16, 2004.

[124] Björn Wittenmark, Johan Nilsson, and Martin Törngren. Timing problems in real-time control systems. In *Proc. of the American Control Conference*, pages 2000–2004, Seattle, Washington, USA, 1995.

[125] Y. Yamada. Exposure tool strategy for 90nm 65nm production. *Semiconductor Fabtech, 18th edition*.

[126] S-M Yang and E-L. Kuo. Damping a hybrid stepper motor with estimated position and velocity. *IEEE Transactions on Power Electronics*, 18(3), 2003.

[127] T. C. Yang. Networked control system: a brief survey. *IEE Proc.-Control Theory Appl.*, 153 (4):403–412, July 2006.

[128] Yuequen Yang, De Xu, Min Tan, and Xianzhong Dai. Stochastic stability analysis and control of networked control systems with randomly varying long time-delays. In *Proc. of the 5th World Congress on Intelligent Control and Automation*, pages 1391–1395, Hangzhou, China, June 2004.

[129] E. Yu and J. Mylopoulos. Why goal oriented requirements engineering. *Proc. 4th Int. Workshop Req. Eng: Foundations of Software Quality, Pisa*, pages 15–22, June 1998.

[130] Mei Yu, Long Wang, Tianguang Chu, and Guangming Xie. An LMI approach to networked control systems with data packet dropout and transmission delays. In *Proc. of the 43rd Conference on Decision and Control*, pages 3545–3550, Atlantis, Paradise Island, Bahamas, December 2004.

[131] By Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21:84–99, February 2001.

Embedded Systems
INSTITUTE

Boderc: Model-based design of high-tech systems

A collaborative research project for multi-disciplinary design analysis of high-tech systems.

Summary

The Boderc Project is an industrial-academic research project, managed by the Embedded Systems Institute in Eindhoven, The Netherlands. For a period of five years, researchers and engineers from the companies Océ-Technologies, Chess IT and Imtech, have worked closely together with researchers of the Technical University Eindhoven, University of Twente, Radboud University Nijmegen and the Embedded Systems Institute. The research objective was to develop multi-disciplinary, model-based techniques and methods for early design exploration of high-tech systems.

Océ-Technologies, a leading international company for professional printing and document management solutions, has provided the industry-as-laboratory environment to develop and validate the proposed embedded systems engineering techniques and methods.

This book summarizes the findings of the Boderc project. Special attention is given to models and methods that predict and analyze system performance, with regard to power consumption, printing throughput, timing, cost-performance trade-off and quality. It provides in-depth insight into the relationship between design choices and resulting system behavior. Furthermore, it creates the basis for multi-disciplinary system design by improving the understanding of the various system level trade-offs and interactions that exist between the mechanical, electrical, and software components. The methods and techniques have been applied to the industrial design practice of high-tech systems, with significant benefits for system quality, costs and time-to-market.

Embedded Systems Institute

The Embedded Systems Institute is a public-private partnership founded in 2002 by a number of Dutch universities and companies: Delft University of Technology, Eindhoven University of Technology, University of Twente, TNO, ASML, Océ and Philips. Its mission is: 'To advance industrial innovation and academic excellence in embedded systems engineering'. A broad research program is executed, mostly based on cases from industrial practice. The research is carried out in partnership with academia, industrial partners and other research institutes. ESI also maintains an extensive knowledge dissemination program, including a wide variety of courses for system architecting and engineering.

Visit us at **www.esi.nl**