An Integrated Approach towards Model-Based Mechatronic Design

AHSAN QAMAR

Licentiate Thesis
Stockholm, Sweden, 2011
Academic thesis, which with the approval of the Royal Institute of Technology, will be presented for public review in fulfillment of the requirements for a Licentiate of Technology in Machine Design. The public review is held in Room B319, Department for Machine Design, Royal Institute of Technology, Brinellvägen 85, Stockholm on 2011-06-08 at 13:00.

© Division of Mechatronics, Department of Machine Design, KTH-Royal Institute of Technology, 2011, Version 20110628132000

Print: US-AB
Abstract

Mechatronic design is an enigma. On the one hand, mechatronic products promise enhanced functionality, and better performance at reduced cost. On the other hand, optimizing mechatronic design concepts is a major challenge to overcome during the design process. In the past, less attention has been paid to the life phases of a mechatronic product, and it was assumed that modifications in electronics and software will ensure that the product performs to expectation throughout its life time. However it has been realized that introducing design changes in mechatronics is not easy, since it is difficult to assess the consequences of a design decision, both during the design process of a new product, and during a design modification. It is also realized that there is a strong need to consider the product’s life phases during the early phases of product development. Furthermore, it is rather difficult to perform a design optimization since it requires introducing changes across different domains, which is not well supported by the methods and tools available today.

This thesis investigates the topic of mechatronic design and attacks some of the major challenges that have been identified regarding the design of mechatronic products. The goal is to provide support to the designers to facilitate better understanding of the consequences of their design choices as early as possible. The work also aims to provide support for assessing alternative design concepts, and for optimizing a design concept based on requirements, constraints and designer preferences at the time of design. The thesis highlights three main challenges related to mechatronic product development: the need for a common language during conceptual design; the inadequate information transfer between engineering domains; and the difficulty in assessing the properties of competing mechatronic concepts. A model-based integration approach is presented, and these key challenges are considered in relation to an integrated modeling and design infrastructure. The approach is illustrated through the design of two mechatronic systems—a two degrees-of-freedom robot, and a hospital bed propulsion system. Initial results provide evidence of good potential for information transfer across mechatronic domains. Although SysML was used for the case studies, some important questions were raised about its suitability as a common language for mechatronics.

Suggestions for future work are: to utilize the developed infrastructure and incorporate a capability to model and assess consequences of competing design concepts; provide support for optimizing these concepts; and evaluate the usefulness of the developed infrastructure in a real-world design setting. These efforts should provide ample information to the designer for making adequate decisions during the design process.
Terminology

The terminologies existing in different engineering disciplines are usually understood and treated differently outside the respective discipline. Due to the multi-disciplinary nature of this research, it is vital to be consistent in the use of key terminology. Hence, the reader is urged to have a look at the following terminology before reading the rest of this document.

- **Design**: The activity of designing (but not the final result)
- **Design concept**: The result of a design activity
- **Discipline**: Technical fields such as electrical engineering, mechanical engineering
- **Domain**: Perspective with respect to a technical system, e.g., mechanical, electrical, software perspectives.
  - *Mechatronic product* is a multi-domain product rather than multi-disciplinary product
  - *Mechatronic design* is a multi-disciplinary activity, but it is partly performed by combining mono-disciplinary efforts
- **Inter-disciplinary** is the same as multi-disciplinary
- **Interface**: A common boundary between components, sub-systems, systems, and products, e.g., a physical connection point
- **Interactions**: Behavior that takes place over an interface
- **Module**: A component or a group of components with a defined interface and known interaction abilities
- **Technical System**: A composite of physical elements and their interactions, which receives inputs, and delivers effects to guide and drive a technical process (see Hubka, Andreasen, and Eder, 1988, chap. Introduction)
- **Function**: The duty of a technical system to deliver specified effects at its output (see Hubka, Andreasen, and Eder, 1988, chap. Introduction)
• **Infrastructure**: Facilities and systems serving a technical system or a process (e.g., a design process)

• **Process**: A logical sequence of tasks performed to achieve a particular objective, defining what is to be done (Estefan, 2008)

• **Methodology**: What consists of techniques for performing a task, i.e., what specifies how to perform a task (Estefan, 2008)

• **Tool**: An instrument or means that (when applied to a particular method) can enhance the efficiency of a task, provided it is applied properly and by somebody with proper skills and training

• **Technology**: The interaction of a technical system with a technical process (analogous to the interaction of a tool and a workpiece), or how the effects (output) of a technical system or human being act to guide and drive a technical process or an operation within it (see Hubka, Andreasen, and Eder, 1988, chap. Introduction)

• **Approach**: The method used or steps taken in setting about a task, or a problem

• **Framework**: What provides the user some guidelines, and special tools to work on a task

• **Formalism**: A description of something conforming to a specific form, logics or mathematics

• **Aspect**: A particular part or feature of something (Oxford Online Dictionary, 2011)

• **Dependency**: Something that is dependent on something else (Merriam Webster Online Dictionary, 2011)

• **Relations**: An aspect or quality that connects two or more things or parts as being, belonging or working together (Merriam Webster Online Dictionary, 2011)
Acknowledgements

This thesis brings me one step closer towards the ultimate goal, i.e. the PhD. degree. I thank Allah Almighty for his blessings in getting me this far. And I would like to express my gratitude to the many people who have supported this research, and who have inspired me throughout the work.

Many thanks to Professor Jan Wikander for providing me with this research opportunity and work towards the PhD. degree. His prompt feedback on my work has helped me a great deal. I would like to extend my thanks to Carl During for initiating this project and bringing an industrial perspective to this research, and for his co-supervision. I would also like to thank Professor Martin Törngren for his advice, and inspiration for undertaking research within the mechatronics division at KTH.

I would like to acknowledge Jonas Torry-Smith (Technical University of Denmark), Sebastian Herzig and Professor Christian Paredis (Georgia Institute of Technology) for many interesting and fruitful discussions, and co-publications. I would also like to thank Professor Lucienne Blessing, Professor Mogens Myrop Andreasen, and Professor Christian Weber, who gave me the opportunity to participate in the engineering design research summer school, and provided critical comments on my research methods, research goals and research directions.

I would like to express my gratitude to my colleagues at the mechatronics division for an inspiring and dynamic research environment! Especially, I am grateful to those who participated in the case studies performed during this thesis, and with whom I have had many fruitful discussions. I am also thankful to Abel Roman Gawel and Hamid Shahid for performing interesting work during their Master’s thesis with me. I greatly acknowledge the feedback from Daniel Frede, Daniel Malmquist, Mohammad Khodabakhshian, Fredrik Asplund, Tahir Naseer, Carl-Johan Sjöstedt, and Vicki Derbyshire for the improvement of this thesis.

Finally, I must thank my wife for her endless support during my research, my parents, sisters and friends for believing in me, and for their encouragement throughout my work.

Ahsan Qamar
Stockholm, May 2011
Contents

Terminology v
Acknowledgements vii
Contents viii
List of appended publications xi
Other publications xiii

1 Introduction 1
  1.1 Background .......................................................... 1
  1.2 Problem formulation .................................................. 3
  1.3 Scope of thesis ......................................................... 5
  1.4 Research questions and hypothesis ................................. 5
  1.5 Example problem ...................................................... 6
    1.5.1 Situation 1: Difficulty in assessing consequences .......... 6
    1.5.2 Situation 2: Inadequate information transfer between domains 7
  1.6 Thesis overview ..................................................... 9
  1.7 Chapter summary .................................................... 11

2 Research approach 13
  2.1 Research methodology ............................................... 13
    2.1.1 Descriptive study 1 .............................................. 14
    2.1.2 Prescriptive study 1 ............................................. 17
    2.1.3 Descriptive study 2, Prescriptive study 2, and Descriptive study 3 ........................................................... 17
  2.2 Research type ....................................................... 17
  2.3 Data collection methods ............................................ 18
  2.4 Chapter summary .................................................... 18

3 State Of the art 21
  3.1 Mechatronic design .................................................. 23
## Contents

3.1.1 Method of investigation ........................................... 23  
3.1.2 Challenges in mechatronic design .............................. 23  
3.1.3 Solutions proposed to overcome challenges in mechatronic de-
    sign ........................................................................ 26  
3.2 Integration ................................................................. 32  
3.3 Consistency .................................................................. 34  
3.4 Conclusions on literature review ................................. 35  
3.5 Chapter summary ......................................................... 37  

4 An integrated approach to mechatronics ....................... 39  
4.1 Case study 1 on robot design ........................................ 39  
4.1.1 Summary of conclusions from the case study 1 ............ 46  
4.2 Case study 2 on hospital bed propulsion system design .... 49  
4.2.1 Conceptual design phase ......................................... 52  
4.2.2 System-level modeling ............................................. 53  
4.2.3 Summary of conclusions from the case study 2 ............ 55  
4.3 The integration infrastructure ....................................... 56  
4.3.1 Integration framework ............................................. 56  
4.3.2 Tool adapters ......................................................... 59  
4.3.3 Meta-models .......................................................... 59  
4.4 Case study 3 on robot design ........................................ 61  
4.4.1 Conclusion on integrated modeling and design infrastruc-
    ture ......................................................................... 65  
4.5 Consistency problems in mechatronic design models ....... 67  
4.5.1 Reflection on consistency problems during model-based en-
    gineering design ....................................................... 70  
4.5.2 Conclusion on consistency management ...................... 72  
4.6 Chapter summary ......................................................... 72  

5 Future work ................................................................. 75  
5.1 Descriptive study 2 ...................................................... 75  
5.1.1 Optimization ........................................................ 76  
5.1.2 Common language issue ......................................... 76  
5.2 Prescriptive study 2 ...................................................... 76  
5.3 A potential descriptive study 3 ..................................... 77  
5.4 Chapter summary ......................................................... 78  

6 Conclusions ................................................................. 79  
6.1 Communication between mechatronic domains ............... 79  
6.1.1 Consistency between mechatronic design models ...... 80  
6.2 Common language issue .............................................. 81  
6.3 Assessing consequences of alternative design concepts ... 82  

Bibliography .................................................................. 85
A Designing Mechatronic Systems, a Model-Based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration 97

B Integrating Multi-Domain Models for the Design and Development of Mechatronic Systems 105

C A Mechatronic Design Infrastructure Integrating Heterogeneous Models 123

D Designing Mechatronic Systems: A Model-Integration Approach 131

E Mechatronic Design- Still A Considerable Challenge 145
List of appended publications

• Paper A

  Ahsan wrote the paper, and developed the concepts for the case study. Jan and Carl provided feedback and corrections.

• Paper B

  Ahsan performed the case study on robot design, and wrote the paper. Martin provided essential feedback for improvements. Jan and Carl provided corrections.

• Paper C

  Ahsan performed the case study on integration of design models for the robot system, and wrote the paper. Jan and Carl provided essential feedback for improvement of the paper.

• Paper D
Ahsan Qamar, Jan Wikander and Carl During, Designing Mechatronic Systems: A Model-Integration Approach, Accepted for Publication in *18th International Conference on Engineering Design (ICED11)*, Copenhagen, Denmark, 2011.
Ahsan coordinated the bed case study with a group of 6 engineers, analyzed the information, and wrote the paper. Jan and Carl provided suggestions for improvements.

- **Paper E**
  Ahsan and Jonas both coordinated the development of concepts for the paper. Ahsan wrote sections 3.3, 4, 5.2 and 5.3, and developed all the visual aids. Jonas wrote sections 3.2 and 3.4, 5.1. Jonas also contributed with the information about the case study. The remaining sections were mutually developed. Jan Wikander, Carl During, Sofiane Achiche, and Niels Henrik Mortensen provided essential feedback for improvement of the paper.
Other publications


Chapter 1

Introduction

1.1 Background

A vast majority of products today are a combination of mechanical, electronic, and software components. This trend is evident in a wide variety of products such as home appliances, construction equipment, automobiles, and airplanes. In terms of engineering such products, the word *mechatronics* is used to differentiate them from purely mechanical products. In general, mechatronics represents a product with components from mechanics, electronics, and software domains, where certain capabilities can be achieved through intelligent design and use of these components. For example, a capability to control the mechanical system. Therefore, in agreement with Torry-Smith and Mortensen (2011), control engineering is regarded as a competence needed to design mechatronic products, among other competences that might be required for different mechatronic products. Mechatronics aim at systematically building up a harmonized product leveraging advantages from each of the involved disciplines.

One can ask the question: why should we develop mechatronic products? The answer lies in the fact that some systems today can only be realized as a mechatronic product, e.g., a CD player, a digital watch, a robot, and a combustion engine (for an automobile) meeting emission legislation. In order to enhance the functionality and performance of a traditional mechanical system, engineers in the early days of mechatronics tried to utilize electronics and software for realizing a number of additional product functions. This trend has increased to such an extent that a large number of functions which were previously realized mechanically are now realized in electronics and software. This provided an advantage of modifying (enhancing) the behavior of a product through software and hardware modifications throughout the product life cycle, e.g., combustion engine control in an automobile through software modification of the fuel injection system. Furthermore, systems which were previously developed independently of each other are now developed as a coupled system in order to provide new product features, e.g., brake and steer by
wire systems. This gives rise to a high complexity of the systems themselves, hence a higher complexity in developing them.

In agreement with Hansen (2000), it can be said that the decision to select a mechatronic alternative for a product as compared to a purely mechanical alternative is based on three dimensions. The first one is attractiveness, i.e., performance properties in terms of user preferences, the second one is economy, i.e., cost for the user, and the third one is tractability, i.e., ability to proceed from design concept into a physically realizable design within allocated resources and time. Among these dimensions, a mechatronic product is also driven by dematerialization and compactness demands. The realization of functions in electronics and software rather than mechanics is a major driver for dematerialization. Dematerialization also aids in creating lighter and more compact products. Therefore, the intended benefits from mechatronics an engineering designer aims for are:

- Cost Efficiency (performance related to cost)
- Dematerialization
- Flexibility to a change through out the life phases (without changing the underlying mechanical systems)
- Eco-friendliness e.g., Eco-cars
- Better control
- More functions inside a product

In order to attain the benefits of mechatronics, large resources are allocated to development of electronics and software sub-systems. Braun and Lindemann (2007) explain that the additional costs in mechatronics as compared to mechanical products stem from: the augmented development activities (in different domains); cross-domain coordination; additional tests and integration processes; higher costs of spare parts in case of a product failure; and new technologies for production and verification. This is also reflected by a decrease in development and production cost for mechanics (from 83% to 28%), as well as a corresponding increase in development and production costs for electronics and software (Braun and Lindemann, 2007). Although it is an aim to develop mechatronic products at less overall cost as compared to traditional mechanical products, problems during development, production, and testing of these products may actually lead to a cost much higher than expectations. Therefore, it can be said that on the one hand, mechatronic products bring benefits, on the other hand, they also bring a complex and more challenging product development. It is vital to address the challenges in developing mechatronic products in order to develop and produce them in a cost-efficient manner.
1.2 Problem formulation

The term mechatronic design is generally used when the mechanical engineering, electrical engineering, and software engineering disciplines interact during the development of a product. A similar statement is given by the German VDI-guideline (Association of German Engineers (VDI), 2004), which states mechatronics as the synergistic integration of mechanical engineering, electrical engineering, and computer science. Mechatronic design is characterized by strong cross linking of involved domains, resulting in complex product development. Within this thesis, the design relating to mechatronic products is also referred as mechatronic design and the product is referred as a mechatronic product. The term mechatronics is used as an umbrella term covering the whole subject area. In agreement with the ideas presented in (Paper E), it can be said that the challenges in mechatronic design stem from the following:

1. Nature of the product: A mechatronic product is multi-domain product by nature. The product is complex in itself due to the large number of functions it performs (Braun and Lindemann, 2007; Tomiyama et al., 2007; Gausemeier et al., 2009a). Furthermore, while designing, each function could be realized as a complex combination of mechanics, electronics, and software components (Gausemeier et al., 2001). This complexity of the product itself, and of the integration required among multi-domain components make it difficult for design engineers to understand a mechatronic system as a whole. It also makes it difficult to assess the consequences of different design concepts in order to evaluate them (Tomiyama et al., 2007; Danilovic and Browning, 2007; Gausemeier et al., 2008; Buur, 1990; Salminen and Verho, 1989).

2. Nature of design activities to develop the product: Design of mechatronic systems is a multi-domain activity by nature. Within each domain, design activities are performed in order to realize the corresponding means to support a mechatronic function. However, since this function is a complex realization of multi-domain components, the design activities within one domain affect what is performed in the other domains (Tomiyama et al., 2007). This is referred to as dependencies. These dependencies make it difficult to synthesize a mechatronic product. It is also difficult to assess the product consequences that are attached with different design concepts. Due to the dependencies between domain-specific design activities, it is important to hunt for means to adequately manage information transfer between domains (Woestenenk et al., 2010). In addition to management of inter-domain dependencies, attention must be paid to relationships among the different components that interact to achieve a desired system behavior. This behavior relies heavily on the interactions between individual components over the defined interfaces (Gausemeier et al., 2001). These relationships and interfaces between multi-domain components need to be carefully treated and discussed while performing mechatronic design, in order to avoid design integration problems.
3. **Background of engineers involved in the design activity:** Engineers involved in designing mechatronic products are typically from different backgrounds. They are usually experts within one domain, hence, it is difficult for them to understand all cross-domain design issues (Salminen and Verho, 1989; Danilovic and Browning, 2007; Buur, 1990). Moreover, design methods, modeling languages, tools, and terminology vary going from one domain to another (Adamsson, 2004). This creates a communication problem between domain experts, which must be solved in order for people to understand not only each other’s design concepts, but also the system as a whole (Frey et al., 2009). Engineers involved in mechatronic design do not only discuss about domain-specific design concepts, but also the system as a whole. It is important to provide some common means of communication to the designers in order for them to better communicate and understand each other.

4. **Nature of development organization for mechatronic products:** The nature of an organization developing mechatronic products influences the design process. As stated by Tomiyama et al. (2007), and Braun and Lindemann (2007), companies utilize cross-disciplinary teams consisting of different domain experts in order to design mechatronic products. With experts from different domains comes the communication issue, as discussed above. Domain specific development is usually carried out not by cross disciplinary teams, but by domain experts. The resulting design concepts thus need to be treated together to facilitate integration. The dependencies thus play an important role and need to be managed appropriately in order to realize a mechatronic concept (Gausemeier et al., 2009b).

The challenges in mechatronics can be grouped based on whether they relate to the product, the development process, the competence of individuals, or the organization (Paper E). This diversity among challenges requires different levels of support. For example, management of dependencies to assess the consequences of selecting between design alternatives, or a common medium for designers to represent a mechatronic concept. In addition to that, an inadequate information transfer between domains slows down the design process as a whole, and hence information transfer needs to be addressed. It has been discussed for long that there is no broadly accepted methodology for mechatronics, and that such a methodology is required in order to address the mentioned challenges. However, since no methodology has gained a wide acceptance within the mechatronic community in the last twenty years, one has to think: what if it is not possible to develop a mechatronic methodology? This is an important question, which should be kept in mind while looking for solutions to overcome challenges in mechatronic design. The reader is referred to Paper E for further details about the mechatronic challenges.
1.3 Scope of thesis

The research proposed in this thesis is mainly focused on how to better develop mechatronic products. In this way, this research is primarily used as a basis for understanding the complex nature of mechatronic products in order to provide better support for designing them. It is also important to investigate how to solve the communication problem between different domain experts while performing mechatronic design. Given the challenges discussed in section 1.2, this thesis will focus towards (challenges) 1, 2, and 3. The challenges related to nature of organization are important, however to limit the scope of this thesis, they will not be treated further. All in all, the research efforts in this thesis aim to provide support for answering some of the key questions (stated in the following) a designer faces while designing mechatronic products:

- How to manage dependencies between domain-specific design activities?
- How to understand complex relations between components in a mechatronic product?
- Is it possible to assess consequences of choosing between different design concepts in mechatronics?
- How to find the best possible design solution while performing design activities in different mechatronic domains?

The research questions in the following section are derived from these key questions.

1.4 Research questions and hypothesis

Based on the problem statement and the scope of this research, the following research questions are important to answer in the research project of which this thesis is a part.

- RQ1: What kind of dependencies exist between different domains in a mechatronic design context?
- RQ2: How can the dependencies be managed while performing design activities in different domains?
- RQ3: Given that there are dependencies between design activities, how to ensure consistency between different design models?
- RQ4: What kind of support can be provided to designers (involved in mechatronic design), that aids them in managing the design activities across different domains, and assists them in dealing with cross-domain design issues?
- RQ5: What is the feasible/adequate method to optimize a design concept given a multi-domain product?
The following hypotheses aim to answer the above research questions

- **Hyp1**: Describing/modeling relationships between design models in different domains during a mechatronic design activity provides a basis for management of mechatronic dependencies (addresses RQ1, and RQ2)

- **Hyp2**: Integrating design models utilized in different domains, supports in ensuring the consistency between models (addresses RQ3)

- **Hyp3**: An infrastructure based on integrated modeling and design supports better management of a mechatronic design activity, reduces development time, and aids designers in managing cross-domain design issues (addresses RQ4)

- **Hyp4**: Performing a cross-domain model-based optimization using an integrated modeling and design infrastructure (for mechatronic design) supports in finding the best possible mechatronic design concept (addresses RQ5)

### 1.5 Example problem

In order to better reflect on the problem statement and the aimed intended support, this section will provide a visual representation of the problem for a simple mechatronic system. The objective is to gain a better understanding of the nature of the problem, and explain how we intend to provide solutions to it. The section is inspired by the visualization exercise proposed by Mogens Myrup Andreasen as explained in (Blessing and Chakrabarti, 2009). A simple two degrees-of-freedom robot is chosen as an example system as shown in figure 1.1. The robot consists of two mechanical arms driven by a motor each, and a sensor that measures the amount of rotation. In the following, different situations are presented, where problems are faced while designing the robot.

#### 1.5.1 Situation 1: Difficulty in assessing consequences

Electronics that are used to drive the motors is packaged as a unit. It is not clear where to best place the unit and which technology to use for the communication. Several solutions are possible, spreading across different mechatronic domains. The unit can be placed on one of the arms (figure 1.2(a)), which would then affect the load balancing on the mechanical structure. However it has advantages of minimizing the wiring and simplifying the cooling of electronics (by using the mechanical structure as a cooling element). It also influences the reliability of the product. Another solution is to place the unit away from the robot, which adds extra cost of the wiring and extra components for cooling. Instead of wiring, a wireless solution can be undertaken. This adds costs of wireless transceivers, but provides advantage of placing the unit away from the robot, gives better protection, easier maintenance, and could be advantageous if the robot is used in a closed environment. However,
a wireless solution introduces constraints on the robot controller, which was originally designed to control over a wired network. On the other hand, it is not clear whether benefits are more than the disadvantages.

The situation as above is commonly faced by designers involved in designing mechatronic products. Since a product concept is usually spread across different domains, it might introduce consequences in the domain-specific designs which were not accounted for earlier. Such situations demand support to the designers, e.g., in the form of an infrastructure, which enables them to understand the effects of the domain-specific design concepts, and put the whole system under consideration rather than in terms of the domain-specific parts. The thesis proposes an intended support based on a common system modeling language to represent a mechatronic concept, and an integrated modeling and design infrastructure (see figure 1.3(b)), which aids in managing relations between the domain models, in order to assess the consequences of different design concepts.

1.5.2 Situation 2: Inadequate information transfer between domains

The development of mechanics, electronics, and software of the robot takes place in the corresponding domains. The design activities performed in these domains affect each other, due to dependencies. For instance, the workspace of the robot is determined by the physical properties, however, the workspace also affects the controller design. The design of software is furthermore based on the processing power of the hardware. The performance of the robot has influences from all
Figure 1.2: Visualized solutions for the hardware unit (a) hardware unit mounted on robot arm, (b) robot unit placed away from the robot with wired connection (c) same as (b) but with wireless connection
Figure 1.3: (a) A visualization of multiple design models affecting a mechatronic product property, namely: controller performance, (b) Visualization of the intended support based on an integrated modeling and design infrastructure, consisting of meta-models for each domain, and a common modeling language (system level model) to represent a mechatronic concept.

domains, e.g., related to motors, sensors, mechanical properties, hardware and software design. It is necessary to provide means for managing information transfer between domains to support the domain-specific development activities. Modeling is an important design activity in mechatronics (Buur and Andreasen, 1989), and a proposed approach to close the gaps between domains is through the integration of domain-models. The idea would be to provide an infrastructure by which the models of mechanics, electronics, software, and control can be integrated in order to exchange information about a property mutually affected by two or more domain models. Figure 1.3(a) shows a visualization, where the property controller performance is affected by the models of mechanics, dynamics, electronics, and software. Integrating these models will provide an ability to exchange information across model and domain borders. For example- the length and inertia attributes between models of mechanics and dynamics, the motor and sensor details between the models of mechanics, dynamics, and electronics. Figure 1.3(b) shows a visualization of the intended infrastructure to support the information transfer between domains. This includes a central model (system model), to which each domain model is integrated by means of the integrated modeling and design infrastructure.

1.6 Thesis overview

This thesis is organized as follows:
Chapter 1 provides an introduction to the thesis, and the problems that are attempted within this research. The important types of challenges in mechatronic design that this research attempts to address are listed to build up the main problem statement. Research questions whose answers are searched for are also stated.

Chapter 2 describes the followed research methodology, i.e., the design research methodology (DRM) (Blessing and Chakrabarti, 2009). The chapter presents details about the research clarification, descriptive and prescriptive studies carried out, with an explanation of the methods used to develop the descriptive and prescriptive studies. The scientific contributions achieved through those studies are highlighted in relation to the research questions drafted in chapter 1. The chapter also classifies the research performed in this thesis among different research types known in design research methodology (Blessing and Chakrabarti, 2009). This helps in clarifying what different stakeholders can expect after the completion of the research project that this thesis is a part of.

Chapter 3 explains one part of the first descriptive study, where the aim is to gain a sufficient understanding of the current situation through literature search performed in relation to areas of relevance for the thesis topic. The chapter highlights the important challenges that are relevant for research in mechatronic design, along with identifying what kind of support is currently available for those challenges. This facilitates completing the reference model, and determining the key factors that might be suitable to address in the prescriptive stage. An initial impact model is also constructed to pinpoint the level of support that is aimed for during the prescriptive study.

Chapter 4 provides further details about the first descriptive study, where the case studies on the robot design and the hospital bed propulsion system design were used to verify the mechatronic challenges identified through the literature review, and gain insights into the gaps in tool support. Later, the details about the first prescriptive study are presented, where the focus is towards development of the intended support i.e., the integrated approach towards mechatronics based on the integrated modeling and design infrastructure. The motivation for this integrated approach, its scope, intended benefits, and limitations are presented. The chapter provides information about the use of integrated modeling and design infrastructure through the third case study performed on the robot design. This facilitates updating the impact model and later evaluate the developed support against the success criteria (identified in the impact model).

Chapter 5 highlights the limitations of the proposed solutions and the suggestions for future work based on a second descriptive study followed by a second prescriptive study. These studies are aimed to evaluate the integrated design and modeling infrastructure in an industrial setting against the measurable success criteria, and incorporate a support for design-optimization along with taking the suggested improvements into account.

Chapter 6 concludes the thesis by summarizing the findings gained, presenting the areas where future efforts will be devoted, and providing inspirations by means of the benefits that could be attained by successful fulfillment of future research
1.7. CHAPTER SUMMARY

In this chapter, we have built a background in the design of mechatronic products. The aims of the research project (of which this thesis is a part) are mentioned to explain the link to the problem area as a whole. The problem formulation facilitated in restricting the scope towards the subset of problems which are attempted in this thesis. This was followed by the formulation of the research questions and the hypotheses, which both play a key role in guiding the research project. We have presented a simple visualization of the problem statement to the reader in order to provide a practical understanding of the challenges faced in mechatronic design.
Chapter 2

Research approach

The research performed in this thesis is aimed to contribute towards building support for designing mechatronic products. In this way, the scientific aim is focused on understanding the challenges the designers face while designing mechatronic products, and providing a tool support for addressing some of the key challenges. These aims are expressed in the form of research questions listed in section 1.4. In the following, the research methodology adopted to find answers to the formulated research questions, and to verify or nullify the chosen hypotheses, is explained.

2.1 Research methodology

The Design Research Methodology (DRM) by Blessing and Chakrabarti (2009) is followed in devising the overall research plan for addressing the stated research questions. The reason for choosing DRM is due to the emphasis of this research on engineering design. The step-wise hands-on approach of DRM describes the ways in which design research can be best undertaken (Blessing and Chakrabarti, 2009). DRM is based on different phases, each devoted to a particular aim. Starting with the research clarification phase, where the aim is to define the main research problem, towards the descriptive study, which is devoted towards understanding the current situation with reference to the research problem. After finding the factors that lead towards the current situation during the first descriptive phase, the areas in which the support will be developed during the prescriptive phase can be determined. Based on the research questions listed in section 1.4, an overall research plan can thus be formulated, following the DRM, as shown in Figure 2.1. The overall research plan consists of a descriptive study aiming to find answers to a few of the listed research questions, followed by a prescriptive study, where the support is developed. In the forthcoming descriptive study 2, the developed support is evaluated, and the feedback is used to devise the prescriptive study 2. This research project consists of two parts, where the first part corresponds to the work performed in this thesis, and the second part corresponds to future work.
Chapter 2. Research Approach

The first part aims to address research questions 1 to 4. However, there is still a potential for future work in order to fully address research questions 1 to 4. This is included as future work, to be performed in the second part of the research project, as explained in chapter 5. Research question 5 is not treated at all in this thesis, but will also be addressed in future work. The research approach followed to answer the research questions is illustrated in figure 2.2. In the following, a short description of each stage within figure 2.2 will be provided.

### 2.1.1 Descriptive study 1

The first stage is focused on analyzing the nature of mechatronic products and the design process that leads to those products. An extensive study of descriptive literature was undertaken to reveal the characteristics of mechatronic products, and the nature of design processes used in developing mechatronic products. In order to develop an infrastructure that aids during mechatronic design, it is necessary to understand what kind of support is required. Descriptive literature such as Buur (1990), Salminen and Verho (1989), Tomiyama et al. (2007), Adamsson (2004), and Gausemeier et al. (2001) describes some of the important challenges that are faced...
2.1. RESEARCH METHODOLOGY

Descriptive literature
Prescriptive literature
Case study in lab

Modeling tools

Descriptive literature
Prescriptive literature

Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Stage 7
Stage 8
Stage 9

Future Work

Figure 2.2: The research approach for the project of which this thesis is a part
during mechatronic design. These challenges relate to: design of the product; the development activities; the mindset of designers; the competence of designers; and the organizational issues (Paper E). In order to understand the consequences of these challenges, a case study was performed on a robot design (documented in Qamar et al. (2010b) and Paper B), and a hospital bed propulsion system design (documented in Paper D). One of the aims of these case studies was to observe the situations, where a challenge results in either adverse consequences for the design of the product, or problems faced during the design activity. This contributes to answering research question 1.

Apart from descriptive literature, prescriptive literature was also reviewed to reveal what kind of support is proposed to address the challenges in mechatronic design. As per the findings based on a thorough literature review performed in Paper E, different types of solutions are proposed in the prescriptive literature. Some of these solutions are: process models adapted for mechatronic design e.g., the VDI-2206 guideline (Association of German Engineers (VDI), 2004); activities based on functional thinking e.g., Buur (1990); Nagel et al. (2008); relationship management e.g., Danilovic and Browning (2007); Felgen et al. (2005); Kreimeyer et al. (2008); integration of design models e.g., Gausemeier et al. (2009b); Paredis and Johnson (2008); Shah et al. (2010); common modeling language for conceptual design e.g., Gausemeier et al. (2009c,a); Shah et al. (2010); Borchers (2011); constraint based design Chen et al. (2009); and requirement driven design e.g., Sage and Rouse (2009). In order to relate the current state of the tool support to the mechatronic challenges, the case studies on the robot and hospital bed propulsion system were extended towards example modeling and design tools, typically used in mechatronic product development. This provided information about the gaps in the tool support, that creates adverse consequences during mechatronic product development, and also helped in studying how to close these gaps with the intended support. It was also investigated during the case studies how the proposed solutions in literature would have helped in addressing some of the challenges. The information obtained through the case studies along with the prescriptive literature provided a partial answer to research questions 2 and 3. The knowledge about mechatronic challenges, and the information obtained from case studies revealed which challenges are not sufficiently addressed through the current tool support (Paper E). This made it possible to define the scope of the intended support (the second stage in the research approach), to be developed in the prescriptive phase.

The third stage aims to study: how the intended support will function during a mechatronic design activity? This includes investigating the following:

1. How the support should help the designer during the design process?
2. What the support should provide to the designer?
3. Which challenges will be addressed by building such a support?
4. What problems still remain (future work or limitations)?
It was decided to focus initial efforts on providing support for information transfer between mechatronic domains, during the different phases of the design process. The resulting support is intended to aid mechatronic design engineers in working on their own design models and in accommodating changes induced by other design models. This will help in providing means for exchanging information across mechatronic domains. However, it was still to be verified whether such a support performs well during the conceptual phase of mechatronic development, where things change at a rapid rate. This point was noted to be checked during the evaluation stage in descriptive study 2.

2.1.2 Prescriptive study 1

Prescriptive study 1 corresponds to the fourth stage in the research approach, and is focused on the development of the intended support i.e., development of an integrated modeling and design infrastructure. Based on the function of this design infrastructure to manage the dependencies between different mechatronic domains, a tool infrastructure is developed. The scope of this infrastructure is limited to example design models from mechanical domain i.e., computer aided design (CAD) and dynamic analysis. System-level modeling is considered as a common language to model systems starting from the conceptual design phase, and leading towards later design phases. The intended support aims to integrate the design models from system-level modeling, mechanical modeling, and dynamic analysis. The support is initially tested through the previously introduced robot design case study, where the aim is to integrate the corresponding design models created in tools: Solid Edge; Matlab/Simulink; and Magic Draw. This provides an initial proof about the functionality of the developed support, along with bringing further answers to research question 4.

2.1.3 Descriptive study 2, Prescriptive study 2, and Descriptive study 3

These studies are included as future work in this thesis. Section 5 provides further details about these studies.

2.2 Research type

This thesis is based on a review-based research clarification phase, followed by the first descriptive study which is also based on reviewing descriptive and prescriptive literature. The prescriptive phase is a comprehensive one where the integrated design and modeling infrastructure is developed. This is followed by an initial evaluation in the second descriptive phase and so on. It can be said that this research aims at development of support and its evaluation (either comprehensive or initial), which is classified as research type 6 in DRM as shown in figure 2.3.
## 2.3 Data collection methods

For the research performed in this thesis, different methods have been utilized to address the formulated research questions and the chosen hypothesis. Figure 2.4 shows the question-method matrix as proposed by Blessing and Chakrabarti (2009, chap. 4). The figure shows which method is utilized to address a particular research question and hypothesis, and if an answer to a research question or a verification or a nullification of a hypothesis can be fully (represented by ‘xx’) or partially (represented by ‘x’) obtained through a particular method. For example, descriptive and prescriptive literature along with case study 1 helps in finding out about the nature of dependencies (RQ1). Whereas, case study 3 fully supports in evaluating hypothesis 1 and 2. Underneath the box showing if the method addresses the research question or hypothesis or not, the effort that is needed to perform a particular method is shown in terms of the researcher (high/low effort represented by ‘RR/R’) and the participant (high/low effort represented by ‘PP/P’).

### 2.4 Chapter summary

This chapter provided details about the overall research plan for the research project (of which this thesis is a part). The research approach (based on DRM) was broken down into different stages, which were distributed among descriptive and prescriptive studies. A short description is provided for the performed descriptive and prescriptive studies. The data collection methods which are used to find answers to the formulated research questions are also presented.
## 2.4. CHAPTER SUMMARY

<table>
<thead>
<tr>
<th>Analyze literature</th>
<th>Case study 1 on example system</th>
<th>Case study 2 on example system</th>
<th>Case study 3 on example system</th>
<th>Assessment exercise</th>
<th>Assessment exercise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptive and prescriptive literature on mechatronic design and optimization</td>
<td>A robot design performed by one design engineer in lab environment</td>
<td>Bed propulsion system design performed by a multi-disciplinary team in lab environment</td>
<td>A robot design performed using integrated design infrastructure by one design engineer</td>
<td>Multi-disciplinary design team in lab environment using the integrated design infrastructure</td>
<td>Designers within a company using the integrated design infrastructure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ1</th>
<th>X</th>
<th>X</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>RR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 2</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>P</td>
<td>RR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ3</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>P</td>
<td>RR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ4</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>P</td>
<td>R</td>
<td>RR</td>
<td>P</td>
</tr>
<tr>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>P</td>
<td>RR</td>
<td>PP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ5</th>
<th>X</th>
<th>XX</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>RR</td>
<td>P</td>
<td>RR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyp1</th>
<th></th>
<th>XX</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR</td>
<td>RR</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>RR</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyp2</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR</td>
<td>RR</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyp3</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR</td>
<td>RR</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyp4</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR</td>
<td>RR</td>
<td>P</td>
</tr>
</tbody>
</table>

---

Figure 2.4: Question-method matrix, showing the data collection methods that can be used to answer the research questions of this research project.
Chapter 3
State Of the art

This chapter begins to describe a part of the descriptive study 1, which is aimed to understand the current situation within the design of mechatronic products, and identify the factors that lead to this situation. The study is review-based, and relevant literature is searched to identify and find the evidence of a problem, or a proposal attacking a particular problem within the design of mechatronic products. The state of the art which is relevant for the research reported in this thesis is shortly described to highlight the gaps that this research aims to partly close. As explained by Blessing and Chakrabarti (2009, chap. 3), it is important to look at disciplines other than one’s own field of research for interesting theories, studies, methods, and concepts that could be relevant. Therefore, a number of potentially relevant areas, including those that are not directly related to the research topic of this thesis were looked at, such as mathematics, psychology, and design space exploration. Figure 3.1 shows the areas of relevance and contribution diagram (ARC). The areas in which the contributions within this thesis are made as well as the areas in which future contributions are expected are highlighted. Figure 3.1 highlights the areas which are not directly related to the topic of this thesis, but are useful to be searched for concepts (marked as useful areas in the figure). Other areas include the ones which are directly related to the topic of this thesis (essential areas), and the ones in which scientific contributions are expected (areas of contribution). Figure 3.1 shows a central ellipse containing the topic of this thesis, i.e., an integrated approach towards mechatronic design. The ellipses that directly connect with this central ellipse are the main subject areas, among which the literature is searched for. These main areas are divided into sub-areas, shown by other ellipses connected to the main ellipse.

The research topic of this thesis can be divided into three main groups: design; mechatronics; and integration. Design research in itself contains many design theories and approaches. Mechatronic products are multi-domain by nature, hence, research in mechatronic design spans many potentially relevant areas. For instance design modeling is an important part of engineering design, but design modeling
Figure 3.1: Areas of relevance and contribution diagram inspired by ARC diagram in Blessing and Chakrabarti (2009, chap.3). The numbers next to green ellipse’s refer to the publications made

for mechatronics spans across different domains, and each domain employs different design methods, modeling languages, and tools. The third area is about integration between domains in the context of mechatronic product development, especially integration between design models created in different domains. Integration between domains has a strong connection to the subject of consistency between models created through design activities performed in different domains. Therefore, integration will also be looked at from the perspective of consistency between design models. In the following sections, the state of the art in mechatronic design, integration approaches, and consistency approaches will be presented. It was tried to search for concepts and contributions within the different areas mentioned in figure 3.1, that are related to mechatronic design, integration, and consistency. In some cases, the state of practice as deduced from the literature review, is also presented.

Based on the reviewed literature, the current situation can be represented through a reference model according to Blessing and Chakrabarti (2009, chap. 4) as shown in figure 3.2. The reference model shows the key factors which play a major role in leading towards the current situation. For mechatronic design, the high level of complexity in integrating domain-specific design concepts is described
3.1. MECHATRONIC DESIGN

3.1.1 Method of investigation

To find the relevant contributions within mechatronic design, a thorough literature study was carried out. The search targets relevant conferences within IEEE (The Institute of Electrical and Electronic Engineers, 2011), ASME (American Society of Mechanical Engineers, 2011), Design Society (The Design Society, 2011), and INCOSE (International Council on Systems Engineering, 2011). Relevant journals within each of these societies were also searched. Apart from this, contributions from mechatronics research groups around the world were read, and shortlisted based on their relevance to mechatronic design.

3.1.2 Challenges in mechatronic design

The first step performed in the literature search was to compile a list of statements regarding challenges in mechatronic design. The list of challenges was studied further through a case study to figure out if there are certain challenges that are still not discussed in the available literature. Figure 3.3 shows the mechatronic challenges and the researchers/research groups stating them (based on the literature review). Further details are available in Paper E.

From figure 3.3, it can be noted that the challenges are grouped as per the following categories:
Figure 3.2: Reference model. The plus sign reflects a positive effect, such as an increase in complexity. The minus sign reflects a negative effect, e.g., a lack of availability.

- **Product**: The challenges related to design and development of a mechatronic product
- **Activity**: The challenges the are faced due to the different types of domain-specific activities performed in designing a mechatronic product
- **Mindset**: The challenges that are faced due to different mindset of designers having a different background
- **Competence**: The challenges related to competence of designers in terms of their education, their communication ability.
- **Organizational aspects**: The challenges that appear due to structuring within an organization developing mechatronic products
- **Other aspects**: This includes other commonly observed challenges related to complexity in developing mechatronic products, and methods used in developing them
### 3.1. MECHATRONIC DESIGN

#### Challenges

<table>
<thead>
<tr>
<th>Category</th>
<th>#</th>
<th>Challenges</th>
<th>Researchers/Research Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>A</td>
<td>Lack of a common understanding of the overall system design</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Difficulty in assessing consequences of selecting between two alternatives</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Lack of a common language to represent a concept</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Modeling and controlling multiple relations in the product concept</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Being in control of the multiple functional states of the product</td>
<td>X X X X X</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Transfer of models and information between domains (expert groups)</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>Activity</td>
<td>G</td>
<td>Synchronizing development activities</td>
<td>X X X X</td>
</tr>
<tr>
<td>Mindset</td>
<td>H</td>
<td>Different tradition within the domains for how to conduct creative sessions</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Reluctant to interact with engineers from other disciplines</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>Different mental models of the system, task, and design related phenomena</td>
<td>X X X X X X X X</td>
</tr>
<tr>
<td>Competence</td>
<td>K</td>
<td>Lack of common language to discuss freely at creative meetings</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>Education within disciplines do not call for integration in professional life</td>
<td>X X</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>The nature of design is different</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Organizational aspects</td>
<td>N</td>
<td>Product complexity affects the organization complexity</td>
<td>X X X</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>Knowledge transfer between domains is inadequate</td>
<td>X X X</td>
</tr>
<tr>
<td>Other Aspects</td>
<td>P</td>
<td>Lack of a broadly accepted methodology</td>
<td>X X X X X X X X</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>Mechatronic ownership is lacking</td>
<td>X X X</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>System engineers are lacking detailed information of the system</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>Complexity as a generic problem</td>
<td>X X X X X X X</td>
</tr>
</tbody>
</table>

#### Figure 3.3: Mechatronic challenges and researchers stating them

The most commonly reported sets of challenges are primarily related to the way a product concept can be described and how information linked to a product concept can be shared across different mechatronic domains. Based on the findings of Paper E, the commonly observed challenges are the following:

- A lack of common understanding of the overall system
- A lack of common language to represent a concept
- A lack of common language to discuss freely

As stated by many of the researchers, the fundamental reason leading to the many challenges is the absence of a common mechatronic design methodology. This is again rooted in the fact that theories building upon different axioms cannot easily be joined to a common theory, as described by Tomiyama et al. (2007).
By observing figure 3.3, it can be said that some of the researchers stand out, since their work is often cited due to their fundamental contribution to design theory. These researchers include Pahl and Beitz (Pahl et al., 2007), Wood (see Nagel et al., 2008), and Suh (2006, 2001). Even though these researchers address mechatronic or complexity issues in their work, a large part of the mechatronic-specific challenges are not discussed by them (Paper E). It is also important to mention the work by a few other researchers for their contributions towards design theories that are relevant for the research performed in this thesis. This includes Tjalve (2003) for his product synthesis model, Hubka and Eder (1988, 1992) for their contribution towards procedural models in engineering design, Tomiyama and Yoshikawa (1987) for General Design Theory, and Andreasen (1980) for the Domain Theory.

Relating to figure 3.3, it should also be mentioned that challenges B, C, and F, and challenges P, R, and S are strongly linked according to the following reasoning. Since there is a lack of a broadly accepted methodology (P) in mechatronics, a common language to represent the concepts can be difficult to accomplish. This lack of a common language contributes to a problem of finding the most suitable design concept through cross-domain design efforts. Along with difficulty in assessing consequences, the lack of methodology and the lack of a common language contribute to a higher design complexity (S) in mechatronics. In addition to that, the lack of common language and inadequate information transfer between domains are strongly connected to challenge R (system engineer lacking detailed information on the system).

### 3.1.3 Solutions proposed to overcome challenges in mechatronic design

Based on the information about challenges in mechatronic design, the solutions to overcome those challenges as proposed in the reviewed literature are investigated. When there is a sufficient documented evidence that a certain proposal addresses one or several challenges in design of mechatronics, then that proposal is considered as a solution, leading towards a list of solutions as shown in figure 3.4. The solutions in figure 3.4 are categorized against the challenges that they address. Each solution is marked as either a partial fulfillment (P), or complete fulfillment (Y). A general overview of figure 3.4 shows that there are several mechatronic challenges which are not sufficiently addressed by the proposed solutions. Specifically, solutions for challenges B, C, F, G, K, M, N, O, P, R, and S are either only partially covered, or no solution is proposed. Among these challenges, the challenges K, and M, relate to competences of designers; challenges G to design activities; and challenge N, and O, to organizational aspects. In order to limit the scope of the research performed in this thesis towards the product related challenges, challenges G, K, M, N, and O will not be considered further. Moreover, as per the earlier discussion, where

---

1Wood is a co-author with Nagel, Stone and McAdams
Figure 3.4: Solutions proposed in literature against the challenges identified in section 3.1.2. Green cells indicate the primary aim of the solutions, and the blue columns show the most important challenges.

relationships between challenges B, C, F and challenges P, R, and S were explained in section 3.1.2, the forthcoming argumentation will be restricted to the challenges B, C, and F, which we believe are at the heart of the mechatronic challenges. These challenges are highlighted in blue in figure 3.4 and in figure 3.3. In the following, a short discussion on each solution listed in figure 3.4 is presented, in order to understand how well challenges B, C, and F are addressed.

Challenge: Difficulty in assessing consequences of selecting between two alternatives

During design, it is common that a designer is faced with a situation where one of the alternative design concepts have to be selected among a few. Such a situation could happen very early in design, or later in a situation requiring a design change on the original design. One such scenario is explained in (Paper E), where a solution to the power consumption problem leads to alternatives like redesign of electronics, software improvement, or battery change (life phase), and it is difficult to weigh one alternative over the other due to insufficient know-how of the effects and the design effort required for each alternative concept. This is a typical characteristic of mechatronic design, where each design concept is spread across different domains.
The main challenge is that there are different ways of solving a design problem, and in mechatronics, it is not always possible, to reason in the best possible way, about the consequences of selecting one alternative over another.

From figure 3.4, it can be seen that several solutions have been proposed in the reviewed literature to help in assessing consequences of a design concept. One of the proposed solutions is relationship management techniques such as Design Structure Matrix (DSM)/Domain Mapping Matrix (DMM) (Braun and Lindemann, 2007; Kreimeyer et al., 2008; Danilovic and Browning, 2007), QFD (Hauser and Clausing, 1988), and FunKey Architecting (Bonnema, 2008). These approaches aid in modeling relationships between functions, and components of a product, along with taking user preferences into account. DSM/DMM have been extended towards multiple layers of functions, components, physical structure, and resources, in order to aid in understanding the complex relations between the design concepts of each domain during mechatronic product development (Braun and Lindemann, 2007; Kreimeyer et al., 2008). Even though these relationship management approaches are useful to gain understanding of multiple relations across domains, it becomes too cumbersome to analyze the consequences of a design choice through them, due to the effort required and efficiency of the method (Paper E).

Other approaches that can be useful for the assessment of consequences include informal description of a design concept such as A3 overviews (Borches, 2011; Borches and Bonnema, 2010b), semi-formal specification language (SFSL) for mechatronics (Gausemeier et al., 2001), and using formal modeling such as SysML (Object Management Group, 2010a) as explained by Shah et al. (2010). These approaches are also developed to aid in providing information to the designer in selecting between different alternatives. However, these approaches only provide a description of a single or few closely related properties or aspects of a mechatronic product. Sketching a design concept (which is rather informal and less restrictive) and revealing its potential advantages and drawbacks has proven to be beneficial in identifying the possible consequences. The case study documented in Paper E supports the argument of sketching being useful. This argumentation for informal and less restrictive modeling leads us to think that A3 overviews should be a possible solution. However, A3 overviews do not support mechatronic specific aspects, such as the implication of different allocation of functions to the domains, as explained by Welp and Jansen (2004). On the other hand, more formal approaches such as SFSL and SysML are good to model different design concepts, and possibly gain understanding about their properties through integration with simulation and analysis languages such as Modelica (Modelica Association, 2010). However, there is a question about their usefulness during the conceptual phase of product development, where a formal approach could cause restriction to the designers in perceiving, and understanding each others design concepts as documented in Borches and Bonnema (2010a). However, using SFSL or SysML, after an initial insight has been gained in the product design can prove to be useful (Paper C; Paper D).
3.1. MECHATRONIC DESIGN

Challenge: Lack of a common language to represent a concept

Mechatronic design is usually performed by a multi-disciplinary team of domain experts, who are unlikely to possess inter-disciplinary knowledge to get a detailed enough understanding of the whole design problem. Therefore, it is difficult to establish a common mechatronic view; rather different domain-specific views are established and the dependencies in-between are unclear. In order to establish a common understanding especially during the conceptual phase of the development, a medium that permits the designers to effectively communicate with each other is required (Paper E). This has been regarded as a common language issue by many researchers within the mechatronic community such as Tomiyama et al. (2007); Woestenenk et al. (2010); Gausemeier et al. (2001, 2009c); Adamsson (2004); Buur (1990); Salminen and Verho (1989).

One of the proposed solutions for a common language is to utilize methods based on functional thinking. Many design approaches based on functional thinking are proposed. These include the functional approach by Buur (1990), the Function-Behavior-State modeling by Umeda et al. (1996, 2005), Function-Behavior-Structure framework by Gero (1990); Gero and Kannengiesser (2004) and Umeda et al. (1990), functional design framework (FDF) (Nagel et al., 2008), and axiomatic design by Suh (2006). Erden et al. (2008) provide a good description of different functional modeling approaches and their applications. However, functional thinking is only a part of the complete picture of the design activities, and other factors such as structural consequences, and how a combination of parts effects one system property are not supported. Functional modeling is abstract in terms of the level at which the description of the product concept is performed. Therefore, it serves the purpose of being a common language, however, only to an abstract level. It is typical after the functional modeling that the development process becomes domain-specific.

Managing the design activities through requirements is proposed as a possible solution to the common language issue. Examples include systems engineering process (Sage and Rouse, 2009), and the work performed by Woestenenk et al. (2010). Requirements can be used for goal specifications (of the product to be), or result specifications (documenting the finalized product). However, in our opinion, the requirements are used to control a design concept, and requirements cannot be used to represent a design concept, which is the sole purpose of the common language.

A holistic approach towards modeling based on informal and formal representations is proposed by different groups of researchers as a solution to the common language issue. An example of informal description is the A3 architecture overviews by Borches and Bonnema (2010b); Borches (2011). This approach aims to provide an overview of the complete system architecture in terms of different system aspects, such as functional and physical aspects. All these aspects are represented on an A3 size paper, hence providing limited but useful information that the designer needs to not only represent his design concept, but also to be able to communicate with other domain experts during a multi-disciplinary design process (Borches,
However, compared to functional thinking approaches, A3 overviews has the same potential of becoming a common language as the functional approaches have, and hence carry the same drawbacks, because the complexity in mechatronic design is neither fully exposed, nor is it manageable through such approaches.

In line with a holistic design approach, a language more specifically related to mechatronics is the *Semi-Formal Specification Language* (SFSL) by Gausemeier et al. (2001, 2007, 2010, 2009c, 2008). This language is proposed to specify a mechatronic concept in terms of a number of aspects such as function, and active structure. It is argued in the proposal of SFSL that a mechatronic system needs to be described through different views, hence leading to different aspects inside SFSL. However, we believe that such a representation will also lead to domain-specific concretization during initial design phases, as also communicated by Gausemeier. SFSL proposes different semantics related to each mechatronic aspect, with the aim that engineers with different background can relate themselves to different aspects supported by SFSL. In that sense, the SFSL is not purely a common language, but can be treated as a language targeting different mechatronic aspects during the conceptual design phase. A different approach from SFSL is the *Systems Modeling Language* (SysML) (Object Management Group, 2010a) by the Object Management Group (OMG). SysML is part of the model-based system engineering (MBSE) initiative to support the systems engineering process, where a system is represented completely in terms of its structure and behavior through different SysML diagrams (see Friedenthal et al., 2008, chap. 3). This helps in establishing a holistic approach towards system design through models, and in providing support throughout the engineering design process. Wölk and Shea (2009), and Follmer et al. (2010) explain the usefulness of SysML in the conceptual design phase of product development. This approach is somewhat similar to the SFSL by Gausemeier, however, Gausemeier suggested that SysML is not suitable for representing a mechatronic concept during conceptual phase (Gausemeier et al., 2009b) since it does not cover different mechatronic aspects, hence leading to the development of SFSL. SysML and SFSL differ in the way aspects are treated, and also in the way SFSL aims to support visual sketches during conceptualization phase. However, a proof of supporting informal ways of sketching is not shown in any of the reviewed literature from Gausemeier. On the other hand, Borches and Bonnema (2010a) document a rather contradicting conclusion to both SFSL and SysML, by showing the results from an empirical research performed in Philips. The authors show that the formal modeling approaches such as SysML, do not usually solve the communication problems between people with different background, nor do they produce models that are easy to understand. It can be concluded here that the approaches such as SysML, SFSL, and A3 overviews can all prove to be beneficial in eliminating many of the problems faced during the current state of the practice in conceptual design of mechatronics. However, these languages do not serve the complete purpose of a common system-level modeling language for mechatronic design during the conceptual phase of the development process.
3.1. MECHATRONIC DESIGN

Challenge: Transfer of models and information between domains (expert groups)

The multi-disciplinary teams involved in designing mechatronic products carry out domain-specific development in order to move forward in the design process, leading to creation of different domain-specific design models, or domain specific views. It is necessary to establish some means of communication between such views in order to avoid integration problems later. Frey et al. (2009) classifies the communication between two design domains in terms of person-person, method-method, model-model, and analysis tool-analysis tool communication possibilities. A person-person communication is regarded as error prone, and method-method communication is not always possible due to lack of analogy between two completely different methods within different domains. Analysis tool-analysis tool communication is possible, however, it is based on execution (e.g., co-simulation) of developed models and not their development. A model-model communication can be considered as possible means of communication between two domains, provided the assumption that what is required to be communicated is captured and available through models. The communication problem might not be that big provided a broadly accepted methodology for mechatronic design gains acceptance. The VDI2206 guideline (Association of German Engineers (VDI), 2004) has been proposed as a mechatronic design methodology. However, it does not discuss how to manage the dependencies between mechatronic domains, nor does the guideline talk about means to address the problem of abstraction between design models and their evolution during design iterations. This further explains the need for model-model communication approaches.

Several approaches targeting integration of two domains through integration of design models are presented in the literature. Creating and using a model as the main artifact during design is the sole aim of the model-based design approach. Therefore, model evolution has to be supported as the development process proceeds. Two dimensions of model evolution has been proposed by Biehl (2010, chap. 4), a horizontal model evolution, and a vertical model evolution. A horizontal dimension of model evolution involves changing the meta-model that the model conforms to, such as evolution of model from one domain to another. A meta-model of a model describes the abstract syntax that a model must follow (see Biehl, 2010, chap. 2). The vertical model evolution includes operations on models such as adding, deleting, or changing model elements (see Biehl, 2010, chap. 4). An important property related to a design model is the level of abstraction. As stated by Buur and Andreasen (1989), a design model is carefully created to only model the product properties necessary at the current design stage. During different stages of the design process, design models are developed at different levels of abstraction. The level of abstraction according to Biehl (see 2010, chap. 2) is defined by the amount of different questions that can be answered through a model. During early design stages, the knowledge about the product is limited, and limited questions can be answered through models that exist at a high level of abstraction.
As the knowledge about the product increases (progressing through different design phases), models at lower levels of abstraction are developed so that further questions about the product properties can be answered. The abstraction between models needs to be supported in order to deal with the complexity, that otherwise cannot be exposed and dealt with.

Based on the two dimensions of model evolution, two types of integration approaches are presented in literature. The first one targets integration of design models existing within different domains. Engelson et al. (2003) showed the use of domain-specific Modelica (Modelica Association, 2010) libraries to perform multi-domain dynamic analysis and controller design tasks on a mechanical system, represented by computer aided design (CAD) models. Bhattacharya et al. (2006) provide an example of integration between CATIA (Dassault Systems., 2011) and Modelica. An example of integration between electrical-CAD (ECAD) and mechanical-CAD (MCAD) models is shown by Chen and Schaefer (2007). The second type of integration approaches involve a vertical model evolution, such as the integration of UML (Object Management Group, 2010b) and Simulink (Mathworks, 2010) models as proposed in (Hooman et al., 2004; Shi et al., 2007; Sjöstedt et al., 2008; Vanderperren and Dehaene, 2006; Brisolara et al., 2007; Boldt, 2006); the integration of SysML (Object Management Group, 2010a) and Modelica models as proposed in (Johnson et al., 2007, 2008; Schamai et al., 2009); the integration of SysML and ECAD models e.g., Shah et al. (2009). Recently, it has been proposed to utilize the system-model for establishing the relationships between the domain-specific design models. This potentially means raising the level of abstraction by going from a domain-specific model towards a system model. To support the evolution of models across different domains, different domain-specific models are raised in abstraction towards the system model, and the system model is utilized to establish relationships between the domain-specific models. Approaches that utilize SysML to establish those relationships include the work by Shah et al. (2010), Paper C, and Paper D.

3.2 Integration

The topic of integration between domains in the context of mechatronic product development has been mentioned as a considerable challenge by many researchers within the mechatronic community (Paper E). Following a model-based development process, a suggested solution to achieve integration between domains is through the integration of design models. Since models are created and modified throughout the design and development process, it is necessary to provide means to support evolution of models as the design process proceeds through different design phases (Paper D). The section (above) on integration of models and information between domains presented a few examples that support model evolution. The evolution of models and the change in abstraction is necessary if a model is to contain only the required level of detail, and nothing more than that. Holt and Perry (2008,
chap. 3) explain that many of the hidden complexities within a system become visible going from one abstraction level to another. Therefore, holistic approaches focus on raising the abstraction level, and modeling the system at a higher abstraction level from the beginning. System-level modeling using SysML is an example of modeling a system in an abstract fashion. The higher abstraction models provide advantages of less domain-specific terminology, ease of modeling a design concept with limited information, and ease of understanding while discussing design concepts with others. In software engineering, many research efforts have focused on creating detailed models automatically by formally describing the abstract models. An example is the code generation from UML (Giese et al., 2009; Giese and Wagner, 2009). In a similar way, many modeling and simulation languages are able to generate code from abstract models. For example, code generation from Simulink (Mathworks, 2011a), code generation from Modelica models through supporting tools (Dassault Systems, 2011b; OpenModelica Consortium, 2011). A similar approach is applied in Computer Aided Manufacturing (CAM), e.g., rapid prototyping machines which can create a product from a computer aided design (CAD) model. Recently, the evolution of SysML has paved the way for utilizing advantages similar to those of UML, including possibilities to embed further capacities by automated transformation towards other design models.

The complexity in designing mechatronic products exists both at the higher abstraction level, where requirements have to be modeled and maintained, and at the lower abstraction level, where detailed design is performed based on simulation and analysis in different domains (Paper A). Due to the focus of SysML to create an information model of a system, many research efforts aim to augment the descriptive capability of SysML with simulation and analysis capability of other modeling languages. Huang et al. (2007) explored the potential for SysML as a modeling tool to support simulation, by modeling in SysML, and creating a mapping for creating a simulation model in eM-plant (Tecnomatrix, 2011), a simulation and analysis tool. Jobe et al. (2008) investigated how to integrate heterogeneous models in SysML using a Multi-Aspect Component Model (MAsCoM) framework. This approach classifies the system in terms of different aspects (representing stakeholder objectives), modeled formally in SysML. Analysis models are formally linked to a particular component-structure model, formally characterized by multiple aspects in an aspect taxonomy. This aids in modeling the relationships between the component models in MAsCoMs to promote model reuse.

Two different approaches towards integration include co-simulation and integration based on a common execution language. Co-simulation relies on exchanging information between two models at run time, e.g., integration between UML and Matlab/Simulink as explained by Vanderperren and Dehaene (2006) and Hooman et al. (2004). An example of a common execution language is GeneralStore (Reichmann et al., 2004). However, these two approaches are only useful to execute the models already developed, and not so useful during the design process where models change dynamically.

Other integration approaches target either providing execution capability for
executing SysML models, such as ParaMagic (InterCAX, 2011) which aids in executing SysML parametric diagrams based on composable objects (Peak et al., 2007a,b), or they focus on integrating SysML with other modeling and simulation languages. Many of these efforts focus on creating a mapping between the two languages. Kerzhner and Paredis (2011) explain using SysML as a framework for relating the system, the requirements, the analysis and the testing. The examples of integration between UML/SysML and Simulink (Brisolara et al., 2007; Hooman et al., 2004; Shi et al., 2007; Sjöstedt et al., 2008; Boldt, 2006) are already discussed earlier in section 3.1.3. Other efforts tried to develop a UML/SysML profile in order to close the gap between abstract semantics of SysML and other modeling languages. Such a profile contains stereotypes applied to UML/SysML elements in order to cope with additional information (extending the meta-model). For example the UML\textsuperscript{H} profile by Nytsch-Geusen (2007) aims for graphical description of Modelica models in UML/SysML. In this case, a combination of UML\textsuperscript{H} editor with a Modelica tool supporting code generation automatically generates Modelica code from UML\textsuperscript{H} model. Other examples of profiles include ModelicaML profile by Pop et al. (2007), a SysML profile for modeling equation based behavior. Schamai et al. (2009, 2010) show examples of integration between SysML and Modelica using ModelicaML. Other examples of integration between SysML and Modelica are by Johnson et al. (2007, 2008), and Shah et al. (2010). An OMG effort aims to standardize the SysML to Modelica transformation (Paredis et al., 2010).

3.3 Consistency

Due to the dependencies between design activities performed in different domains in mechatronic product development, an issue of consistency between the corresponding design models arises. This is due to the fact that two or more design models can affect one product property, and hence the relationships between models’ elements and the corresponding changes within each model has to be consistent with what exists in the other models. As is the case with the model transformation approaches, most of the work focused on consistency issues between design models originates from the software engineering discipline. The topic of consistency between mechatronic design models has recently gotten some attention within the mechatronics research community, and different software engineering methods are utilized by researchers to solve the problems in mechatronic design. The types of consistency issues that are dealt with in literature can be split into two categories. The first category deals with consistency within a model itself (Herzig et al., 2011). Approaches such as Mens et al. (2005) and Simmonds et al. (2004) are examples, where authors propose to use a description logic to satisfy the dependencies between model elements created through languages such as UML or SysML, in order to ensure consistency within a model itself. However, the presented work is not complete and it still needs further developments in order to be proved beneficial.

The second category in the reviewed literature deals with consistency issues
3.4. CONCLUSIONS ON LITERATURE REVIEW

across multi-domain models, that are created through different modeling languages. Adourian and Vangheluwe (2007) show how to relate two different modeling languages through their corresponding meta-models. The approach is similar to the model integration approaches discussed in section 3.1.3, where the language constructs of each language are related to each other (at the meta-level). An association meta-model takes care of relationships between model elements of both the languages. A bi-directional transformation is then used to ensure consistency between models developed through both the languages, such as between Solid Edge (Siemens PLM Software, 2011b) and UML as shown by the authors. Gausemeier et al. (2009b, 2007) explain a similar approach for managing consistency between a domain-spanning principle solution, and domain-specific models through triple graph grammars (TGG) (Schurr, 1994). Automated model transformations from the principle solution to domain-specific models support the evolution of models during transition from conceptual phase towards embodiment, and detailed design phases. However, for both the approaches by Adourian and Gausemeier, modifying or deleting individual components could still lead to inconsistencies among models.

An approach for instant consistency checking based on a model analyzer approach is shown by Hehenberger et al. (2010). Any change to a design model performed by the user is observed by the system that triggers a consistency checker, and evaluates the relevant consistency rule for the performed changes. The authors have argued that this approach provides support for maintaining consistency between models developed across different domains. However, it is difficult to define the consistency rules for all possible scenarios, and some consistency rules can only be defined in an abstract manner. Although this approach is useful, it can be said that it still solves a subset of the consistency problems that arise during the mechatronic development process, and the topic of consistency management needs further attention.

3.4 Conclusions on literature review

The comprehensive literature review in the descriptive study 1 leads to the following conclusions:

- Different approaches have been proposed to aid in mechatronic design. However, many challenges related to the design of the product are still not addressed. Especially, the challenges related to: assessing consequences of different design alternatives to aid in decision making; the common language issue during conceptual design of mechatronic products; and the information transfer between domains require further research efforts to be addressed.

- Different approaches have been proposed for integration between domains. They are based on different system-level modeling approaches. Typically, a gap between mechanical design and electronic design has been observed during the design of mechatronic products. The integration approaches consider the
mechanical design to stand still, however, in the aim of finding the best design concepts during mechatronic design, liberty has to be provided to designers in order to easily modify the mechanical design, even after the initial conceptual design phase.

- Different approaches to ensure consistency between models have been proposed, but they are mostly incomplete. This topic is getting further attention within the mechatronic community, due to the topic’s strong connection to design problems in mechatronics. However, there is still a lot of room for development in this area.

Based on the above conclusions from the literature review, it is possible to make an initial decision about the potential areas in which research efforts will be made during the prescriptive study 1. A good way to visualize this is through the development of an initial impact model as shown in figure 3.5. This initial impact model shows the intended support, and its aims to change the existing situation depicted by the reference model in figure 3.2. This initial impact model will be revised after determining the characteristics of tool support through a case study (documented in chapter 4). The aim is to identify the gaps in the current tool-support for modeling and design of mechatronic systems, so as to define the specific focus of the intended support.
3.5 Chapter summary

This chapter provided details about the part of the descriptive study 1, where the current situation was deduced through the literature review. The state of the art relating to mechatronic design, integration between mechatronic domains, and consistency between design models is presented. Based on this literature review, the reference model was developed to determine the factors that lead towards the current situation in mechatronic product development. A list of challenges in mechatronic design was developed, based on the review of the descriptive literature. Among these challenges, a few of the key challenges related to product development were explained. A further review of the prescriptive literature provided information about which solutions were proposed to overcome the key challenges, and which areas required further research. These investigations aided in deciding the areas in which support will be developed during the prescriptive study phase, and led to the creation of the initial impact model.
Chapter 4

An integrated approach to mechatronics

This chapter explains the three case studies, which are a part of the descriptive study 1, explaining the problems that are to be resolved, followed by the prescriptive study 1, where solutions to the problems are developed and applied. Two mechatronic example systems are discussed during the case studies. The design of a two-degrees-of freedom robot as one example, and the design of a hospital bed propulsion system as the second. All the case studies are performed in a laboratory environment, hence, the solutions proposed carry evidence from the laboratory environment only. Further studies in an industrial setting will be performed in future.

4.1 Case study 1 on robot design

This case study focuses on the design of a two degrees of freedom robot. The robot is used as an example mechatronic product, keeping in mind the complexity of the robot as a product, and the complexity in designing it. Different design solutions for the robot are known from the literature and authors’ experience. The knowledge about these solutions allows us to focus on the problems faced due to the capabilities of the current tool support while performing the robot design. As a result, it is possible to gain insight into the gaps in current tool support. This complements the information about mechatronic challenges identified through the literature review (see section 3.1.2). Knowing the characteristics of the tool support and the characteristics of mechatronic design, this case study helps in defining the focus of the intended support (see figure 2.2), which is explained in section 4.3. This case study was performed by a single person with a background in mechatronics. Therefore, the case study does not mimic a real situation where people with different backgrounds collaborate. Hence, the case study is limited to reveal the problems in the tool support, and not the problems that exist due to communication between designers. The later will be considered in the next two case studies.
The case study is performed in order to make a practical observation of the three shortlisted challenges related to design of mechatronic products (see section 3.1.3). These include the following:

1. Difficulty in assessing consequences of selecting between two alternatives
2. Lack of a common language to represent a concept
3. Transfer of models and information between domains

The reviewed literature suggests that the Systems Modeling Language (SysML) (Object Management Group, 2010a) can be utilized as a common language, and can be used to model alternative design concepts, which can be analyzed through support from other simulation and analysis tools. To further study this proposal, and to understand the challenge related to the availability of a common language for conceptual design of mechatronic products, SysML is used while performing the robot design. In this way, the effectiveness of SysML during conceptual design can be analyzed, along with establishing how to support system-level modeling during different design phases. While creating a SysML model of the robot, it can also be determined how to support analysis of different design concepts through a SysML model, with the objective to aid the designer in determining the consequences associated with a design concept. It is important to build upon the understanding gained in the conceptual design phase while transitioning towards embodiment and detailed phase. In model based development, the information gained through this understanding can be represented through models. While it is possible that domain-specific activities are performed during the conceptual design phase, the embodiment and the detailed design phases are typically supported through domain-specific activities. Therefore, it is necessary to investigate how to transfer models and information between domains, and how to keep domain-specific models consistent with the evolution of a model developed to represent the complete system, e.g., a system model developed using SysML. In line with these thoughts, this case study aims to provide answers to a few important questions (presented in the following) which are derived from the research questions presented in section 1.4:

- What kind of integration (within and across domains) exists in current tool support, and what gaps still remain due to limited integration among tools?
- How can the common language issue be addressed for the conceptual design of mechatronic products, and what are the characteristics of a common language?
- How to integrate design models for information transfer across domains?
- Is it possible to assess consequences of alternative design concepts through SysML modeling?
In the following, a short overview of the case study is provided, followed by a summary of conclusions gained through it.

The main function of a two degrees-of-freedom robot is to navigate in a planar workspace, in the presence of obstacles. The robot design begins with putting down the requirements that the robot needs to fulfill. As per Tjalve’s model of a product synthesis process (Tjalve, 2003), the requirements lead to the creation of functions and identification of means to satisfy these functions. The functions/means lead to creation of basic structures for a product. A further selection and refinement from the basic structures leads to quantified structures. All this is performed against the design criteria, which is used throughout the design process (Tjalve, 2003).

For the robot case study, a quantified structure is shown in figure 4.1. Based on an initial consensus to continue with this quantified structure, the system-level modeling is performed to model the robot requirements, the functions the robot needs to perform, the structural aspects, and the behavioral aspects of the robot. Enterprise Architect (Sparx Systems, 2011) and Magic Draw (No Magic, 2011) are two different tools (among many others) supporting SysML modeling, providing different user interfaces, and additional plug-ins. The robot model was initially created in Enterprise Architect, and the details can be seen in Qamar et al. (2010b). Later, the same model was created in Magic Draw, which provided further insights into the differences among SysML modeling tools. Figure 4.2 shows a model of the robot requirements, which in figure 4.3 are followed by a creation of a use-case diagram showing a functional view of the robot. Based on the functional view, and the knowledge about quantified structure of the robot, a structural view of the robot is created as shown in figure 4.4. The structural view can be broken
CHAPTER 4. AN INTEGRATED APPROACH TO MECHATRONICS

Figure 4.2: Top level requirements for the robot

Figure 4.3: Robot Functional View represented through Use Case diagram
Figure 4.4: Top level structure of the robot. The emphasis here is more towards mechanical parts, however, electrical components such as actuator drivers, CAN bus, and sensor electronics could also be modeled.

Figure 4.5: Internal block diagram showing mechanical assembly.
down to further limit the scope to mechanical assembly (figure 4.5) and control configuration (figure 4.6).

Different design criteria are chosen for the robot design such as maximum controlled position error, response time, and cost. A number of analysis contexts are described in the SysML model, each related to a certain design property. The position context shown in figure 4.7 is related to the calculation of end-position of the robot, for analyzing the controlled position error. Similarly, other contexts such as hardware performance, and software execution contexts are built. It is important to note here, that these contexts aim to gain an understanding about a property of a system, whose realization can be seen as a combination of mechanics, electronics, and software components. For example, the maximum controlled position error depends upon mechanics (work space of the robot), hardware performance, software execution, and the design of the controller. Hence, choosing alternative concepts for the hardware, or for mechanics may require complete rework of the mechanical design or electrical design (domain-specific designs) in order to gain insight about
maximum position control error (a product property). These situations require information transfer between domains, which can be established by integrating the design models created in each domain. There is an underlying assumption in this approach, which relies on the fact that the information that needs to be transferred in between domains is available in a model. However, during a design process, a designer makes many decisions, and it is possible that the rationale behind these decisions is not explicitly documented in a model (see Biehl, 2010, chap. 4), but this topic is beyond the scope of this thesis. Hence, the forthcoming sections will be presented provided the assumption about information availability inside a design model is true.

![Diagram showing the position context of the robot for calculation of end-effector position](image)

Figure 4.7: Defining the position context of the robot for calculation of end-effector position as represented by EndEffectorPosition in the PositionContext block

In order to understand the controlled performance of the robot, the mechanical design of the robot was performed in Solid Edge, a mechanical CAD tool (see figure 4.8(b)). As documented in Paper D, the multi-domain nature of a mechatronic design requires design models that are capable of modeling multi-domain physical systems. Modelica, MapleSim (Maplesoft, 2011), Simscape toolbox in Simulink
(Mathworks, 2011b) are different possibilities to model a multi-domain physical system. For this case study, the Simscape toolbox is utilized, and a SimMechanics model of the robot (figure 4.8(a)) was constructed based on the information known from the CAD model of the robot (figure 4.8(b)). There are many views that can be created while modeling the robot system, however, we have only looked at a few of these views, and some views such as an electrical CAD view are not treated in this thesis to simplify the investigation process. The SimMechanics model helps in understanding the dynamic behavior of the robot, along with gaining information about the performance of the designed controller on the mechanical system. The SimMechanics model is a behavior model of the robot, but behavior aspects of the robot were also modeled in the SysML model, such as state based behavior (figure 4.9). While it is possible to model the behavior aspects of the robot in SysML, their possible execution and refinement are done through domain-specific tools, such as Stateflow for state-based behavior in Simulink. Development of this kind of models also requires an integration of models, in order to provide a capacity to analyze while designing, hence supporting the decision making process.

![SimMechanics model and CAD model](image)

**Figure 4.8:** (a) Robot SimMechanics model (b) Robot CAD model in Solid Edge

### 4.1.1 Summary of conclusions from the case study 1

From the case study performed on the design of the robot, the following conclusions can be stated:

- The case study showed that the challenges list compiled through the literature review covers the major challenges faced during mechatronic design, i.e., no further challenges were observed during the case study apart from the ones formulated in the challenges list.

- By performing an investigation about the gaps in current tools, the level of integration that exists among tools within a domain, and the level of integration existing among tools across domains could be determined. This investigation
showed that in general, there is a lack of integration between mechanical design and electrical/electronic design tools. The same type of gaps exist due to a lack of integration between software design, electrical/electronic design, and mechanical design tools. Most of the tools were developed to support a domain expert during design and analysis within a particular domain, hence such gaps in the tool support are understandable.

- A certain level of integration exists in between tools within a particular domain. For example, interoperability between mechanical CAD tools, interoperability between electrical CAD tools.

- There are ongoing efforts within research and by the tool providers to reduce the gaps in tool support. Such efforts include for example Mechatronic Concept Designer (Siemens PLM Software, 2011a), and CATIA V6 (Dassault Systems, 2011a). These efforts aim to reduce the gaps within a tool chain from one tool vendor, and they do not necessarily eliminate the gaps if other tools are utilized.

- It is possible to model a mechatronic system as a whole in SysML, however, this has to be critically judged against:

  - The need to educate engineers to understand SysML models.
  - The understanding of a design concept represented through a SysML model by a group of people with different background.
– The effort needed to support system-level modeling throughout different design phases (in contrast to system engineers not aware of domain-specific development).

– The amount of effort involved in creating the SysML model. It is possible that a SysML model becomes too complex, hence not serving its purpose.

– The need to execute a SysML model for simulation and analysis. A SysML model is not executable since the main purpose is to create an information model of the system. Hence, the execution capacity is left to other modeling and simulation tools.

• SysML is in an evolution phase, and so are the SysML tools. Some of the tools are rather difficult to use, hence, it is not a straight-forward procedure to obtain full benefits from using them.

• The conceptual phase of design is rather dynamic, and it is not clear whether a restriction to formal modeling like SysML aids in the design process, or leads to a lack of creative thinking while formulating design concepts. Therefore, we deduce that it is feasible to begin with the system-level modeling after the design concepts for basic structures and quantified structures of the product have been found. This is treated further in case study 2 as explained in section 4.2.

• If system level modeling is performed, it should be supported throughout different phases of product development. This requires integration of design models developed during the conceptual design phase with the design models developed during domain-specific development activities.

• A considerable gap is observed because of a lack of integration across domains, while utilizing design tools during domain-specific development activities. This hinders the design activity, since modelers must rely on time-consuming and error-prone (inter-disciplinary) discussions in coming towards a design solution. It is necessary to provide means for integrating models developed in domain-specific design tools, in order to provide better information flow across domains.

• It is not straight-forward to assess consequences of selecting between two design alternatives, even while utilizing SysML and creating a system model. Although, alternative design concepts can be modeled in SysML, and different solutions have been proposed in literature to perform analysis on the alternative design concepts, however, these solutions are still not complete. For example, mechanical design is assumed to be frozen, which is not an ideal case. Furthermore, it is not enough to only study the product properties that alternative design concepts aim to address. There might be other effects, such as higher design effort due to complete rework required within a particular domain. There might also be consequences for production apart
from the design. Current tools, design methods, and practices are far way from a support level, where consequences of a mechatronic design concept can be systematically analyzed. The conceptual design phase starts with knowing the requirements on the product, however, these requirements continue to evolve, and some requirements will only be known after investigative discussions, such as sketching of design concepts to reveal the life phases of the product. These requirements can not always be revealed and taken into account while only modeling in SysML, and a design alternative may be chosen while overlooking the effects of the product’s life phases which are vital to consider.

In accordance with the deductions of this case study, it is decided to investigate the integration of design models further, as a means to reduce gaps in tool support. However, the integration of models can be divided into two parts- an integration of system level models with domain specific models; and an integration between domain specific models themselves. Section 4.3 will provide further details about model integration.

### 4.2 Case study 2 on hospital bed propulsion system design

Building upon the findings of the first case study, the second case study starts with investigating the means of information transfer across domains. In a model-based development process, a model is the primary artifact, and each model reproduces the properties of the object being modeled (Buur and Andreasen, 1989). As design proceeds through different design phases, models increase in detail. Depending on the design stage, a design model can be abstract or detailed. However, it is important that a design model is carefully developed to only model the product properties necessary at the current design stage (Buur and Andreasen, 1989). This restriction in scope is necessary- firstly, since information about a design problem increases through different phases of design; secondly, because a design model with too many product properties will become unnecessarily complex to serve the purpose of the designer. Therefore, different design models (reproducing different product properties) are utilized in a product development process. In a mechatronic design scenario, design models vary between different design domains. Some of these models define and describe the product from the domain perspective such as mechanics or electronics; others are used to evaluate product properties within a domain such as dynamic analysis of a mechanical design. As the design proceeds, there is a resulting change in abstraction of a model. Therefore, it is necessary to support different abstractions during the modeling process and manage information transfer across mechatronic domains throughout different design phases.

In order to support the conceptual design of mechatronic products, the need for common language has been discussed in the mechatronics community as presented in section 3.1.3. This common language according to Buur and Andreasen (1989) should support:
The abstract function structure independent of technology

Modeling of function principles supported through different technologies

The specification of the interfaces between different technologies

Evaluation of SysML to serve this purpose is explained by Follmer et al. (2010) and Wölk and Shea (2009). This case study will focus on evaluating how to support system level modeling through different design phases in order to close the gap between design models created to understand the system as a whole (during the conceptual design phase), and the design models created during domain-specific development activities. A design study is performed on a servo-propelled hospital bed to answer the following questions:

1. How to establish relationships between domain-independent and domain-specific design models?

2. How to integrate different design models developed at different design stages?

Figure 4.10: The integration infrastructure for model based development

The conceptual design phase is very dynamic in terms of the rate at which design concepts change and in terms of interaction of designers. Therefore, supporting
the development process through formal models during conceptual design phase requires a good ability to change the models easily and rapidly. For this reason, it is suggested in this thesis to start with the development of the system model after performing initial product synthesis as shown in figure 4.10. Figure 4.10 is developed in agreement with Tjalve’s model of the product synthesis process (Tjalve, 2003), starting with the problem formulation and leading to creation of quantified structures of a product. Developing a system model after identifying a few of the competing quantified structures for a product will support in modeling the product as a whole, and performing simulation and analysis to gain an understanding about different properties of the product. For this reason, it is also an aim of this case study to analyze whether it is possible to support an assessment of consequences of alternative design concepts by modeling them in SysML, and performing analysis through integration of SysML with other modeling and simulation languages.

<table>
<thead>
<tr>
<th>Criteria/Aalternatives</th>
<th>Manufacturing cost</th>
<th>Development cost</th>
<th>Ease of maneuver</th>
<th>Safety</th>
<th>Ease of installation</th>
<th>Configurability</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighting factors</td>
<td>0.3</td>
<td>0.1</td>
<td>0.25</td>
<td>0.05</td>
<td>0.15</td>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>Diagonal</td>
<td>4/1.2</td>
<td>4/0.4</td>
<td>4/1</td>
<td>4/0.2</td>
<td>4/0.6</td>
<td>1/0.15</td>
<td>3.55</td>
</tr>
<tr>
<td>Straight vertical</td>
<td>4/1.2</td>
<td>4/0.4</td>
<td>3/0.75</td>
<td>3/0.15</td>
<td>4/0.6</td>
<td>1/0.15</td>
<td>3.25</td>
</tr>
<tr>
<td>Straight horizontal</td>
<td>4/1.2</td>
<td>4/0.4</td>
<td>3/0.75</td>
<td>3/0.15</td>
<td>4/0.6</td>
<td>1/0.15</td>
<td>3.25</td>
</tr>
<tr>
<td>Three</td>
<td>3/0.9</td>
<td>5/0.5</td>
<td>4/1</td>
<td>4/0.2</td>
<td>3/0.45</td>
<td>1/0.15</td>
<td>3.2</td>
</tr>
<tr>
<td>Four</td>
<td>2/0.6</td>
<td>5/0.5</td>
<td>5/1.25</td>
<td>5/0.25</td>
<td>2/0.3</td>
<td>1/0.15</td>
<td>3.05</td>
</tr>
<tr>
<td>Configurable</td>
<td>3/0.9</td>
<td>2/0.2</td>
<td>4/1</td>
<td>4/0.2</td>
<td>3/0.45</td>
<td>5/0.75</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Figure 4.11: Decision matrix for six design alternatives of hospital bed. Criteria-lowest=1, highest=5.

In order to provide the means of information transfer through design models, an infrastructure supporting abstraction and evolution of models is required. The system model can serve as a common platform for different domain experts to understand and reflect on the design of the product during the conceptual design phase. The infrastructure shown in figure 4.10 aims to support evolution of a system model while transitioning from conceptual design phase towards embodiment and detailed design phases, and performing domain-specific development activities. This is achieved by a model-level integration between system model and the domain-specific models. The model-level integration is based on specifying the relationships between the system model and a domain model, which in turn aids in relating one domain model with the other domain models. Using SysML, these relationships can be built by utilizing the extension capability in SysML to create domain-specific parts of the system model (figure 4.10), where the parts are built
with concepts of a particular domain. Hence, the system model increases in detail as design progresses through domain-specific design models. Design iterations continue to take place between different design stages, and design models consistently evolve through the integration infrastructure (figure 4.10). Figure 4.10 shows another route (red arrows) where domain-specific design models are created based on the information obtained through the product synthesis process. The infrastructure facilitates information transfer between these domain-specific design models by relating them via the system model through the integration infrastructure.

### 4.2.1 Conceptual design phase

The aim of the hospital bed propulsion system design exercise is to design an active (driven) wheel module, which can be utilized on common hospital beds. During the conceptual design phase, the requirements for the hospital bed design were established. The design team consisted of people with background in mechanical engineering, electrical engineering, control engineering, and computer science. The team came up with different propulsion system configurations, each with a drive and a steer capability. These configurations were compared in a weighted decisions matrix, to find out the most suitable configurations with respect to the design preferences (see figure 4.11).

![Figure 4.12: Main Structure of the hospital bed propulsion system](image)

Figure 4.12: Main Structure of the hospital bed propulsion system
It was concluded that some of the design concepts needed further analysis to evaluate different properties, before any decision about them could be made. For example, the configuration of the wheel module as a combined driving and steering unit is one of the concepts that required further analysis. This lead to the creation of mechanical computer aided design (MCAD) models to further study the wheel module configurations. Two wheel module configurations were discussed and modeled in a MCAD tool. Preliminary analyses were also performed on both these configurations to predict the movement of the center of gravity of the bed while using each configuration. Further details about the MCAD models and the analysis models can be found in Paper D.

Figure 4.13: A model of a trade-off study for propulsion system configuration alternatives. The criteria such as configurability and developing cost are represented as measures of effectiveness (moe) of each alternative.

### 4.2.2 System-level modeling

Knowing the requirements for the design of the wheel module along with the possible solution concepts, a system model was built. The requirements for the hospital bed, the functions that the bed needs to perform, and the structural and behavioral
aspects of the bed were modeled using SysML as a common modeling language during conceptual design. One such aspect is shown in figure 4.12, where the structural view of the hospital bed propulsion is shown. SysML also allows to actively control system design through requirements, along with distributing the visualization of the same system or subsystem into different views, for ease of understanding. It was also possible to represent the alternative design concepts in a SysML model, e.g., the propulsion system configuration alternatives were modeled as a trade-off study as shown in figure 4.13.

The trade-off study can be used to evaluate the properties of the different propulsion system configurations, hence supporting in decision making. It also partially aids in assessing consequences of choosing one propulsion system configuration over another. The assessment is partial since it is not possible to evaluate all product properties through a model during the conceptual design phase. There might also be other factors such as amount of development effort required for a design alternative within each domain, which cannot be deduced through a model, since a model does not usually contain such information. It is also important to consider the life phases of the product while assessing the design alternatives, but the life phases cannot be easily addressed through SysML models. Figure 4.14 shows the two wheel module configuration alternatives, which were analyzed for form and behavior through an MCAD model and a dynamic analysis model. Further details about both of these models, their relationship to the system model, and control configuration for the bed are available in Paper D.
4.2.3 Summary of conclusions from the case study 2

The following conclusions can be drawn from the case study performed on the design of a propulsion system for a hospital bed.

- The case study verified the conclusion from case study 1 that the list of challenges compiled through the literature review indeed covers all the major challenges faced during mechatronic design.

- System level modeling is helpful during the conceptual design phase, especially to visualize and understand the integration of subsystems. SysML aids in modeling a product in terms of different aspects, but they are not specifically related to mechatronic aspects, and they are modeled using general purpose SysML semantics.

- In order to make good use of system level modeling, the system model needs to co-evolve along with the domain-specific models.

- It is possible to represent alternative design concepts through SysML, for further evaluation. However, the evaluation requires integration of SysML with other modeling and simulation languages, which is an active research area. Therefore, it will take some time before tool vendors provide solutions for performing analysis using SysML models. Paramagic (InterCAX, 2011) is one of the currently available tools, that aids in executing SysML parametric diagrams.

- The usefulness of a SysML model can be questioned during the conceptual design, based on a lack of cross-domain understanding gained through it (by engineers with different background), and the amount of effort required to develop and maintain the model.

- Using a formal or a semi formal language during the conceptual design is useful for expressing the information gained during the phase. On the other hand, it limits the independence of a design engineer for expressing his or her design ideas. In particular, informal modeling such as sketching provides better and simpler visualization of concepts. However, it is difficult to extract the information from sketches and represent it in a model for possible reuse and further work.

- For supporting the system level modeling throughout different design phases, an infrastructure that supports information transfer between different design models is proposed. This infrastructure will be used in case study 3 (see section 4.4) to show an example of model integration between different design models.

- It is possible to relate a domain-specific design model with a SysML model by creation of a SysML profile for that domain. In return, this also supports
the establishment of relationships between one domain-specific design model to the other through the SysML model.

![Figure 4.15: Concept of model transformation (Czarnecki and Helsen, 2006)](image)

### 4.3 The integration infrastructure

As discussed earlier, in order to provide the means of information transfer through design models in the perspective of engineering design, an infrastructure supporting abstraction and evolution of models is required. As the design proceeds through different design phases, the dependencies between domain-specific design activities result in relationships between the corresponding design concepts that emerge from each domain. However, the domain specific models do not incorporate these dependencies explicitly, though they have an implicit influence on how these models evolve. It is proposed that each domain-specific model can be related to the system model by creating a mapping between the corresponding meta-models. A mapping is a specification of the correspondences between meta-models, where a meta model describes the abstract syntax that a model must follow (Biehl, 2010). As a result of specifying the meta-model-mappings between SysML and different domain-specific modeling languages, it is possible to utilize a SysML model for establishing relationships between different domain-specific models. Consequently, the SysML model contains both the system description and the relationships to different domain-specific models. This will aid in managing dependencies between design concepts resulting from domain-specific design activities, along with maintaining a complete system view in accordance with the progress achieved through the domain-specific design activities.

#### 4.3.1 Integration framework

The mapping between SysML and domain specific models can be used to create the means of abstraction between the corresponding models. The objective of the
4.3. THE INTEGRATION INFRASTRUCTURE

The integration framework is to provide support for building those means. The aim is to represent the modeling formalisms of different domains in a common formalism, in order to facilitate information exchange between them. Eclipse Modeling Framework (EMF) (Eclipse Foundation, 2009b) is one such platform. Inside EMF, a model is defined as an Ecore model. A meta-model to which the model should conform can be defined inside EMF. By defining the meta-model, EMF allows generation of corresponding plugins in order to create a model that conforms to the meta-model. Model transformations can then be performed in between two models. Model transformation is an automated process of converting a source model into a target model, through defined transformation rules (Czarnecki and Helsen, 2006), where the transformation rules are based on the mapping specification between the two meta-models. The transformation is then executed by a model transformation engine as shown in figure 4.15.

![Diagram](image)

Figure 4.16: (a) Integration infrastructure visualization for one domain, (b) Integration infrastructure visualization for different domains

The integration framework is based on definition of the meta-models for different modeling languages used in different mechatronic domains. A domain model contains information that is relevant for other domain-models. However, a domain-model also contains information that is relevant only within the domain-model itself. By definition of meta-models, it is possible to control the information trans-
fer procedure so as to only extract the information that is relevant for other domains. For a model developed within a domain, this leads to definition of a Domain meta-model as shown in figure 4.16(a). An Ecore model called Domain model in figure 4.16(a) can be created by reading the model from the modeling tool (used within the domain) through a tool adapter.

SysML is a general purpose systems modeling language, and a SysML model is created through general purpose SysML semantics. Therefore, SysML lacks formal and detailed semantics needed to represent the domain-specific information. However, domain-specific semantics can be introduced into SysML through creation of SysML Profiles. A profile extends the SysML meta-model by defining the constructs which relate closely to the concepts within a particular domain, or to the constructs of a domain specific modeling language. This meta-model extension allows performing model transformations to create a domain-specific SysML model (using the corresponding profile). The integration framework inside figure 4.16(a) shows a domain-specific SysML model that conforms to the domain-specific SysML meta-model. A tool adapter for a SysML tool is used to create the target model called Domain model through SysML profile inside SysML (see figure 4.16(a)). Now,
it is possible to build relationships (relation *SysML model-domain model*) between this generated domain model and the system model, which essentially means relating the parts in the system model to the corresponding domain-specific parts. For other domains, the same procedure is applied, starting from a domain model, and leading towards the domain-model created using a SysML profile for that particular domain, as shown in figure 4.16(b).

### 4.3.2 Tool adapters

Completing the specification of meta-models means that a model transformation can be applied to them. During this thesis, two different model transformation languages have been used—the Atlas Transformation Language (ATL) (Eclipse Foundation, 2009a) which is a declarative model transformation language, and the Model Query Language (MQL) which is an imperative model transformation language supported by Cameo Workbench (No Magic and Sodius, 2011). Cameo Workbench is based on EMF, and provides built-in connectors to different modeling tools such as Magic Draw (No Magic, 2011) which is a SysML modeling tool, and Matlab/Simulink. This means that the models developed in these tools can be read as Ecore models using Cameo Workbench.

Different modeling tools are utilized during case study 1 and case study 2 (discussed earlier) including Solid Edge for mechanical CAD modeling, Matlab/Simulink for dynamic analysis and controller design, MapleSim for dynamic analysis, Magic Draw and Enterprise Architect for SysML modeling. Cameo Workbench is utilized to read and write models in Magic Draw and Matlab/Simulink. For Solid Edge, a tool adapter was written in Visual Basic to read a Solid Edge model, or create a model inside Solid Edge. This adapter is able to read a parameterized model inside Solid Edge and write it into an XML file complying to the Solid Edge meta-model. Similarly, it is possible to create a Solid Edge model by reading the XML file. Figure 4.17 shows the integration infrastructure example for integrating a Solid Edge model, and a Simulink/SimMechanics model through a SysML model. Different modeling tools are available to create mechanical CAD (MCAD) models. It is proposed to utilize a generalized meta-model that is applicable for all the tools, since they use similar MCAD concepts. A similar strategy is proposed for electrical CAD tools, and dynamic analysis tools. Hence in figure 4.17, the meta-model is referred as MCAD instead of Solid Edge meta-model and the SysML profile is referred as SysML profile for MCAD. In a similar fashion, a SysML profile for ECAD, or dynamic analysis (DA) can be used.

### 4.3.3 Meta-models

Based on figure 4.17, a meta-model each is defined for MCAD, and Simulink, along with the meta-model relating to SysML profile for MCAD, and SysML profile for DA. Figure 4.18 shows the Solid Edge meta-model, which, in accordance with our proposal, can be utilized as an MCAD meta-model.
Figure 4.18: Solid Edge domain meta-model in Ecore

The meta-model is based on MCAD concepts such as assembly, part, relation. Figure 4.19 shows the SysML profile for Solid Edge (MCAD). Observing figure 4.18 and figure 4.19, a similarity between names of the constructs can be observed, e.g., `cadAssembly` in meta-model for SysML profile for MCAD relates to assembly in Solid Edge meta model. The extension of SysML constructs to create the MCAD constructs (domain-specific constructs) can be seen in figure 4.20, showing the SysML profile for Solid Edge.

The meta-model for Simulink is available in Cameo WorkBench, a snapshot of which is shown in figure 4.21 showing features of a Simulink Block. The SimMechanics toolbox in Simulink provides an adequate representation of the mechanical system. The SimMechanics model is hence read through the Simulink connector inside Cameo WorkBench, which populates an Ecore model complying to the Simulink meta model. A SysML profile for Simulink/SimMechanics (shown as SysML pro-
4.4 Case study 3 on robot design

This case study is aimed at utilizing the integration infrastructure presented in section 4.3 for the robot example presented earlier in case study 1. The aim is to integrate different design models developed during the robot design example, and try to assess if integrating models adequately aids in information transfer across...
domains, and more importantly, assess its benefits for the designer. In the following, the result of integration between the Solid Edge and SysML models of the robot will be shown. The integration between Simulink and SysML is also discussed.

Figure 4.23 shows a visualization of integrating the design models for the robot example using the integration infrastructure. The Solid Edge model of the robot shown in figure 4.8(b) is read through the tool adapter for Solid Edge (Solid Edge Parser). An ATL transformation generates an Ecore model conforming to the SysML profile for MCAD. Cameo Workbench reads/writes a Magic Draw SysML model as a UML 2.1 meta-model with SysML profile. Therefore, it is required to transform the generated model conforming to the SysML profile for Solid Edge into a model conforming to UML 2.1 with a SysML profile, and a SysML profile for MCAD. This transformation is written in MQL. The Magic Draw connector (available inside Cameo Workbench) is then used to create a representation of the Solid Edge model of the robot inside SysML. The complete transformation procedure from Solid Edge to SysML is shown in figure 4.24.

Figure 4.25 shows the generated model after performing the transformation from Solid Edge to SysML. It is now possible to allocate the components from the system model of the robot as shown in figure 4.5 to the generated model from Solid Edge.
Figure 4.21: A snapshot of the Simulink meta-model in Cameo WorkBench (No Magic and Sodius, 2011)
Figure 4.22: Meta-model for SysML profile for Simulink/SimMechanics defined in Cameo WorkBench. The features of SimMechanicsElement can be seen.
Figure 4.23: Visualization of robot example through the integration infrastructure shown in figure 4.25. These allocation relationships are shown in figure 4.26, where the correspondence relations between SysML model and Solid Edge model of the robot are shown.

In a similar fashion as shown in figure 4.24, the transformation between Simulink/SimMechanics and SysML is performed, which is an ongoing work.

4.4.1 Conclusion on integrated modeling and design infrastructure

The following conclusions can be drawn from the use of the integrated modeling and design infrastructure during case study 3

- The infrastructure supports in understanding of relationships between system design and domain specific design models. Consequently, it aids in manage-
ment of dependencies between design activities performed in different domains, during different design phases.

- The procedure is based on the creation of a SysML profile for each domain, which aids in relating the SysML model with the domain-specific model. However, the procedure also means creating a number of profiles for every domain, and maintaining them as the modeling languages evolve. Such an activity might not be economically viable for a company, and should be supported by tool vendors.

- There is a part of information in a model that is relevant for other domains. However, there is also information such as tool environment parameters that need to be managed in order to successfully read/write a model inside a tool. For development of a tool adapter, it is required to take care of this kind of information, which might not be directly relevant from the perspective of transformation between models. This was observed during development of a tool adapter for Solid Edge. Therefore, in our opinion, these adapters should also be supported by tool vendors.

- The infrastructure is only used for small scale models and only within a lab environment. It is necessary to analyze the effectiveness of the proposed
4.5. CONSISTENCY PROBLEMS IN MECHATRONIC DESIGN MODELS

Figure 4.25: Transformation result from Solid Edge to SysML for the robot example

procedure by obtaining feedback from modelers/designers in a realistic design setting.

- In order to ease the transformation process, a meta-model for each domain
  can be utilized instead of a using a meta model for a model created through
  one particular modeling language within a domain. For example, MCAD
  meta-model for MCAD tools, and ECAD meta-model for ECAD tools.

4.5 Consistency problems in mechatronic design models

Consistency refers to a state in which no contradictions are present. Contradictions
could arise from many sources during the design process. During the model-based
design process, a model is treated as a primary artifact. As stated by Buur and
Andreasen (1989), a model reproduces the properties of the object being modeled.
This means that a mapping exists between the model and the object being modeled.
During the design process, the object which is the product, is still an unknown
Figure 4.26: Correspondence relationships between the structural model of the robot and the MCAD model built in Solid Edge

reality. Therefore, different methods are adapted during the design process in order to gain better understanding of the product, e.g., through models, through prototyping, and through testing. All these methods are affected by the decisions the designers make during the design process. This is in line with Tjalve’s (Tjalve, 2003) thoughts, where it is possible that two different quantified structures lead to a different looking end product, however, two products based on different basic structures will be completely different.

The road through which designers find these basic structures, and the quantified structures of the product is based on the information they possess at the current time of product design, their preferences, and beliefs, each of which may lead to contradictions in the product design concepts in relation to the final aimed product. So how to identify if the design concepts that are being modeled are inconsistent with the final product that is visualized, and how to check for these inconsistencies during the design process? To answer such questions, an understanding about what is consistent and what is inconsistent is required, along with information about different types of inconsistencies that can occur during the course of design process (Herzig et al., 2011). Then it can be studied whether it is possible to identify
A model could be a formal or an informal representation of the object. For example, sketching is an informal form of modeling a design concept. Sketching is based on visualization of the concept in the mind of the modeler. Whether this design concept is consistent with the laws of nature and the real world object that is aimed for or not is classified as an external consistency problem (Herzig et al., 2011). External consistency applies to the reality in which the product will ultimately exist. For mechatronic systems, this refers to the real-world, where laws of nature such as physics, biology should be followed. However, our understanding about laws of nature is not perfect, and such an understanding can be treated as a belief about the law of nature and not the actual law itself. Therefore, such a belief is subjected to an amount of uncertainty. Furthermore, the knowledge about the product in design increases as the design process proceeds, and hence the final product is not known before it is actually built. Hence it is possible to have a design concept, that is inconsistent with the final product that is aimed for, and such an inconsistency according to (Herzig et al., 2011) cannot be checked for. It is rather impossible to check for external consistency problems.

Internal consistency problems are related to systems that are developed under certain well understood axioms such as logical and mathematical rules (Herzig et al., 2011). Modeling languages are constructed based on these rules, and the grammar of the modeling language must satisfy the constraints on logic and mathematics. The models developed through these modeling languages should reflect the proper use of the language constructs in order to be internally consistent. Not following the rules of the language hence leads to logical contradictions, which are possible if modeling languages do not strictly enforce the proper use of the rules, i.e. they are not completely formal modeling languages. For example, the semi-formal nature of UML and SysML as discussed in (Broy et al., April, 2010). Therefore, such a nature of a modeling language leads to situations where it is not possible for the tools to enforce development of models conforming to logical rules of the language, and hence not all the logical inconsistencies can be detected (Herzig et al., 2011).

In addition to the logical consistency, failure to adhere to the laws of mathematics while constructing a model will lead to mathematical inconsistencies, such as failure to adhere to Kolmogorov’s axiom of probability. For example, during a fair coin toss, if it is assumed that the probability of heads is 0.5, and probability of tails is 0.6, then the concept is internally inconsistent with laws of mathematics. This can be linked to external consistency, if the probability of heads is 0.3, and of tails is 0.7, hence leading to total probability of 1, not violating Kolmogorov’s axioms and hence internally consistent. But this concept will be externally inconsistent with the law of nature for a fair coin toss, where the observations demand equal probability of heads and tails. A modeling language if not adhering to laws of mathematics, may allow a modeler to build a model to be internally inconsistent such as above, but inconsistent with the laws of nature (external inconsistency).
4.5.1 Reflection on consistency problems during model-based engineering design

In order to further illustrate the concepts of consistency discussed earlier, this section will shortly present the robot design example (as used in case study 1 and case study 3), in relation to the external and internal consistency problems. The design concepts for the robot were modeled using SysML, following the best practices of model based systems engineering (MBSE). Matlab/Simulink and Solid Edge were used to refine the design concepts to aid in the decision making process and reach a principle solution. The robot design started with the initial set of requirements (see figure 4.2) towards creation of the functional view of the robot (see figure 4.3). In order to provide means to perform the intended functions, two quantified structures of the robot were chosen based on a basic structure as shown in figure 4.27.

The quantified structures are evaluated against a set of basic properties (designer preferences) such as reliability, manufacturing cost, and controllability, and the aim is to predict these properties in order to assess the two design alternatives. For example, the reliability of the robot is the product of the reliability of each degree of freedom, which is the product of the reliability of individual components i.e., the arm, joint, motor, and position encoder. The motor and the actuator are typically off-the-shelf components, for which the reliability figures are provided by the manufacturer. For the arm, which for this example is a manufactured component, a stress and strain test can be performed to select a confidence level above 99%. However, for this example, it is clear that the failure of a motor not only leads to the malfunction in one degree of freedom, but makes the whole robot unusable for further operation. Therefore, the reliability of each quantified structure can be predicted based on the corresponding component probability and further study of the design concepts for cause and failure effects. When specifying reliability characteristics of the robot, consistency with kolmogorov axioms, hence the laws of mathematics must be ensured. For example, specifying the probability of the robot not failing to be 99.999%, while declaring the robot to be not able to fulfil its function within the same time frame to be 10%, is clearly an inconsistent specification (Herzig et al., 2011).

Apart from modeling the structural aspects, SysML also allows modeling of behavior aspects of the robot. This leads to different views inside a SysML model. Hence, it is required that all these views are internally consistent with each other, i.e., requirements consistent with function specification, functions with structure, structure with behavior, and behavior with requirements. Furthermore, each of these views within the SysML model should be logically consistent with rules of SysML, i.e., the SysML grammar. Other modeling languages such as Matlab/Simulink and Solid Edge were used to refine the design concepts, and aid in the decision making process. However, as explained earlier, a model influences other models due to the dependency between the design activities performed by modelers in different domains. For example, controllability is dependent upon mechanical
properties of the robot, while the workspace of the robot is dependent upon the physical structure. A mechanical design can satisfy workspace requirement with having a bad controllability, and a controller design may satisfy the controller requirements but with a suggested change in mechanical design. Therefore, in order to be internally consistent, not only the information within the model needs to be consistent, but also the information across different domain-models (Herzig et al., 2011). Unfortunately, it is rather difficult to identify the relations between models as a result of hidden dependencies, and more importantly the effect of those relations is not clearly known, hence they are not considered during the course of the modeling process.

While there are solutions proposed in the literature to perform internal consistency checks, such as consistency checking of UML models (Mens et al., 2005; Simmons et al., 2004), and maintaining consistency across different domain-models such as Gausemeier et al. (2009b); Adourian and Vangheluwe (2007), checking
whether the given system is externally consistent with e.g., laws of nature, or the final product as desired is not possible. Further inconsistencies related to beliefs and preferences of the designer may also exist as explained in (Herzig et al., 2011).

### 4.5.2 Conclusion on consistency management

Based on an initial investigation on the issues of consistency management during the development of mechatronic products, the following conclusions can be drawn.

- **It is not possible to check for external consistency.** Since the final product is unknown, it cannot be checked whether a current design concept is consistent with the final aimed product, or if a design concept is consistent with the laws of nature.

- **Inconsistencies can occur while making decisions during the design process,** such as the ones related to the beliefs and preferences of the decision maker. Such inconsistencies can only be checked if the information related to the design decisions is explicitly available, e.g., through models. This is however not always the case, hence, it is not always possible to detect such inconsistencies.

- **It is possible to check for internal consistency such as logical inconsistencies provided that the model is developed through a formal modeling language,** which is however not the case with UML and SysML.

- **Internal consistency between models developed across different mechatronic domains is an important problem.** Unfortunately, most of the solutions proposed in reviewed literature do not guarantee complete consistency, since modifying and deleting model elements can lead to inconsistency.

- **It is not possible to check for all possible types of inconsistencies.** However, there is a need for further research in this area, especially related to consistency between design models during the development of mechatronic products.

### 4.6 Chapter summary

This chapter enlightened about the first two case studies that were performed as a part of the descriptive study 1. The conclusions gained from both these case studies are summarized, which also lead towards deciding the function of the intended support. Later, the intended support (i.e., the integrated modeling and design infrastructure) which was developed during the prescriptive study 1 is explained. The functionality of this support is explained through an example of integration (i.e., the third case study) between Solid Edge and SysML models, and a possible integration between Simulink and SysML models for a two degrees of robot system. The chapter also presented the concepts of external and internal consistency
Figure 4.28: Impact model revised after the prescriptive study 1

between design models, which were illustrated through the robot design example. Figure 4.28 shows the revised impact model after completing the prescriptive study 1. The impact model shows a negative sign next to the success factor, which means that still no ease is gained in finding the best design concept among a few after building the intended support. This will be taken into account in the descriptive study 2, which is part of the future work.
Chapter 5

Future work

This section presents the future work which will be performed in the next phase of the research project of which this thesis is a part. In the following, the descriptive and prescriptive studies that will be performed as per DRM, and the research questions which these studies aim to answer will be shortly discussed.

5.1 Descriptive study 2

In the prescriptive study 1, the intended support based on a model integration infrastructure was developed (see section 4.3). The developed support was initially applied to the design of the robot, as discussed in the case study 3 in section 4.4. The aim of descriptive study 2 is to evaluate the built support; this evaluation is the fifth stage of the research approach presented in figure 2.2. Initially, the impact of integrations between Solid Edge and SysML models, and between Simulink and SysML models for the robot design will be analyzed. Later, it will be studied how effective these integrations are for managing relationships between Solid Edge and Simulink models, i.e., how effectively the relationships between different design models are managed through the integration infrastructure. The next aim is to perform an assessment exercise in a realistic setting, potentially in industry, or in a lab environment. Here a group of design engineers will design a product using the integrated design infrastructure. The designers feedback will be recorded through interviews/questionnaire, in order to assess whether the integration infrastructure aids the designers sufficiently, and to judge its limitations. It is possible to utilize the bed propulsion system as the design problem during the assessment exercise, and carry on further from the conceptual design, towards detailed design, and building a prototype. The transition from the conceptual design towards the detailed design will provide further inputs about the use of the integrated design infrastructure throughout the different product design phases.

The evaluation performed in descriptive study 2 will help in verifying or nullifying the hypothesis 3 based on the research question 4, i.e., whether the devel-
developed support helps the designers (involved in mechatronic product development) in managing dependencies across different domains, and assists in dealing with cross-domain design issues. Furthermore, the evaluation should also provide information about whether internal consistency between design models could be maintained through the developed support (research question 3), and how efficiently the dependencies between design activities across different domains can be managed through the developed support (research question 2). The evaluation results will help in defining potential areas of improvement for the integrated modeling and design infrastructure, which can be taken into account during prescriptive study 2.

5.1.1 Optimization

With an aim to optimize a mechatronic design (concept), various optimization methods will be studied for a potential implementation as a part of the integrated design infrastructure. This translates towards building additional integrated support during prescriptive study 2, to facilitate optimization of mechatronic designs, i.e., providing an initial answer to research question 5.

5.1.2 Common language issue

The need for a common language during the conceptual design of mechatronic products has been discussed earlier. SysML is utilized during the course of this thesis to serve as a common language. However, there are other system modeling approaches which can be investigated. This thesis leads us to believe that currently different system modeling approaches are useful for different purposes, and in different organizational environments. A comparison of these system modeling approaches will help in identifying which approach is useful under which scenario, along with a discussion on pros and cons of each approach. In agreement with the discussion presented in Paper E, we suggest that an informal sketching carries substantial importance during conceptual design phase, especially in terms of understanding of a design concept by a group of people with different backgrounds. Therefore, it will be beneficial if informal sketching is supported by an underlying modeling and simulation engine, which carries a potential to serve as a common language for the design of mechatronic products. It can be concluded that studying the provision of a common language in mechatronics is an important research problem. If the common language can be realized, it would also facilitate in creating variations of the product concepts during the conceptual design phase, and in presenting the desired mechatronic views to support engineers from different domains (Paper E).

5.2 Prescriptive study 2

This study will be aimed at accommodating the feedback obtained in the descriptive study 2 regarding the integrated design infrastructure. Further additions and modifications on the developed support are expected to be performed at this stage.
5.3 A potential descriptive study 3

The aim of this study would be to evaluate the effectiveness of the proposed support in assessing consequences between alternative design concepts during a mechatronic product design, along with assessing how easy it is to modify a product model while trying to find the optimal design (concept). Hence, it would be possible to judge whether the integration infrastructure and optimization methods are feasible in a real design scenario, where they should simplify the process of finding the optimal concepts for mechatronic products.
5.4 Chapter summary

In this chapter, the future work is highlighted, which will be performed in the next phase of this research project. The future work is mainly based on two studies—the descriptive study 2; and the prescriptive study 2. One of the aims of the descriptive study 2 is to evaluate the integrated modeling and design infrastructure in a real design setting involving a multi-disciplinary design team. This will be performed by carrying on the hospital bed design towards the construction of a prototype, and make observations about how effective the design infrastructure is for the designers. The second aim of the descriptive study 2 is to study different optimization approaches, among which one approach will be implemented during the prescriptive study 2. The objective of this prescriptive study is to incorporate support for optimizing and evaluating product concepts in the integrated modeling and design infrastructure. This is shown by gaining in ease in finding the best design concept while developing mechatronic products as shown in figure 5.1.
Chapter 6

Conclusions

This thesis presented an approach for integrated model-based design of mechatronic products. After initial product synthesis, the design of a mechatronic system typically splits across different domains, hence requiring integration approaches to manage the development process over the whole product life cycle. The state of the art and practice surveys based on reviewed literature suggests that this partitioning into domain-specific development activities leads to substantial gaps between activities performed across different domains. These gaps can also be observed in the tools used in different domains. Inadequate information transfer across mechatronic domains has thus been listed as a major challenge. It is vital to provide a means of information transfer across domain borders to bridge such gaps to be able to smoothly carry on during the design process. By providing the means for such information transfer, it is further possible to better assess the consequences of alternative design concepts. However, there will always be situations where the consequences cannot be assessed because the required information is unavailable.

A wide variety of challenges are faced during the design of mechatronic products, and these alone constitute a field of extensive research. To date, no single methodology providing adequate support for the design of mechatronic products has gained wide acceptance in the mechatronic community. Nevertheless, different solutions do exist to address key challenges such as the common language issue, the information transfer between domains, and assessing consequences of alternative design concepts. However, currently, these solutions are mostly incomplete, leaving room for major development in these areas (Paper E). A short conclusion to these challenges is provided below, based on the work performed in this thesis.

6.1 Communication between mechatronic domains

The thesis presented an integrated modeling and design infrastructure to provide support for information transfer across mechatronic domains. The infrastructure is used for the design of a robot system, and the preliminary results show that
it is possible to relate system-level design with domain-specific design by building relationships between the system model and the domain-specific model. It was possible to raise the abstraction level of a domain-specific model such as transforming a Solid Edge model into a SysML model. There is only a part of each domain-specific model that influences a product property, in addition to influences from other domain-specific models (see figure 1.3). Through the definition of partial meta-models and the transformation rules between them, we can control which part of a source model is read, and what is created in the target model. In this way, the integration between system design and domain specific design provides an ability to maintain both an abstract view and a detailed view of the product. In agreement with Holt and Perry (2008), we believe that such an abstraction also supports revealing the whole complexity which otherwise can not be seen. The main conclusion here is that relationships between different domain specific models can be established through a SysML model, and that maintaining these relationships throughout the design process aids in communication between mechatronic domains.

The results from use of the integration infrastructure during robot design showed that the integration of models is an important step for maintaining information transfer across domains. However, it is not yet clear how effective this approach is in a real-world design setting, and whether it has a potential for adoption by designers working in industry. For this reason, the future work aims to perform a design exercise in a realistic (mechatronic) setting, to find out if the designers actually find the integration infrastructure useful or not, and also to verify the integration between MCAD-SysML and between DA-SysML through a real-world design context.

Revisiting the research questions and hypotheses formulated in section 1.4, hypothesis 1 is verified through the information gained by using the integrated design infrastructure. However a concrete characterization of dependencies is difficult to formulate, suggesting that research question 1 is not fully answered. On the other hand, the proposed way of managing dependencies through the integration infrastructure does provide an answer to research question 2.

### 6.1.1 Consistency between mechatronic design models

One of the aims of the integration infrastructure is to keep the different modeling views of the product consistent with each other. A language like SysML supports modeling different views. However, as documented in Herzig et al. (2011), it is possible that the SysML model is inconsistent in itself, and such an inconsistency is difficult to detect. Consistency between multi-domain models firstly requires each model to be consistent with the rules of the underlying language; secondly, it requires that information about the modeled artifact is consistent among the involved domains (models). The integration infrastructure can support keeping information between models consistent with each other, however, there can be situations which are not supported by the integration infrastructure, and which arise
when adding or deleting operations are performed on a model. Hence it is difficult to claim complete consistency through the use of the integration infrastructure. Defining different consistency rules on top of a product architecture, as suggested in Hehenberger et al. (2010), is a promising way to manage consistency. However, it is difficult to develop consistency rules for all possible product architectures, and hence the approach still contains many open questions.

Revisiting the research questions and hypothesis formulated in section 1.4, hypothesis 2 is not fully verified, and answers to research question 3 are not fully obtained at present. The impact of model changes on multi-model consistency is still an unresolved problem. Hence consistency management will be further studied during the planned future work.

6.2 Common language issue

During this thesis, we have come to a conclusion that there is still no generally accepted language within the mechatronic community that serves the purpose of a common language during conceptual design. The importance of informal sketching during conceptual design is inevitable. However, this makes it difficult to represent the resulting knowledge (gained by informal sketching) in the form of formal models. By interpreting design concepts represented through informal sketches through a modeling and simulation engine, it will be possible to simulate them, which in our opinion can be considered as the future for a common language in mechatronics. SysML was utilized in this thesis to serve the purpose of a common language. It must be noted that SysML is a language evolving with time, and by no means is SysML proposed as the final solution to the common language issue. Instead, this thesis proposes maintaining a view on the complete system through a common language that effectively serves the purpose of designers during the conceptual design phase (where a partitioning of design activities may happen), and which can be used during the transition from conceptual design towards embodiment and detailed design phases. Due to the inherent nature of SysML, a gap is identified between the conceptual design phase and the creation of the system model. It is not reasonable to start only with the SysML modeling from the very beginning of the mechatronic design process, due to the rapid rate of concept creation, and change in design during the initial concept synthesis (being part of the conceptual design phase). Therefore, it is suggested to begin with system-level modeling after initial product synthesis has been performed. It is also recommended to evaluate the use of SysML (to support a model based systems engineering process) based on the level of understanding gained through SysML models by engineers with different backgrounds. It must be kept in mind that utilizing SysML could lead towards modeling being too expensive. Nevertheless, if SysML is utilized, it should be integrated into the complete tool chain used in order to gain complete benefits from integrated modeling.

Utilizing a common language is part of the verification needed for hypothesis 3,
and since this issue is not yet resolved, this part of hypothesis 3 remains unconfirmed at present. The same applies to research question 4, where the common language is a part of the integrated modeling and design infrastructure. At this moment, only a partial answer has been obtained, and further work is required in this area.

6.3 Assessing consequences of alternative design concepts

It is important to predict the properties of alternative design concepts in order to make a decision upon which concept to carry forward. This is rather complex in mechatronic design, since a function is realized as a complex combination of mechanics, electronics and software. It is sometimes difficult to predict the amount of development effort required for each design concept. In other cases, it is rather difficult to take into account the life phases of the product while choosing between design alternatives. In most cases, the life phases of a product are evaluated based on informal sketching (as mentioned in Paper E), which is rather different from formal modeling.

If a design change is required during the product development process, the tendency is usually towards modification of electronics and software, which in some cases solves the problem, but in other cases, it may lead to integration failures. Moreover, this tradition also steers the whole system towards not being optimized, since the underlying mechanical system is kept unchanged. We suggest that if a common language is used during the conceptual design phase, the alternative design concepts should also be modeled and analyzed using the model developed through the common language. For example, when using SysML as a common language, model alternative design concepts in a SysML model, and predict the properties of those concepts through SysML (by utilizing integration with other languages). This should let the designer understand the different properties of a design concept in relation to the preferences of different stakeholders. In contrast to what might be assumed when using SysML, it is important to point out that not all the information required to assess consequences (of a design concept) is present in models, as also explained in Paper E. When assessing the consequences of alternative design concepts and studying the life phases of a product, there are still major differences between using SysML and using informal sketching. This thesis investigated whether it is possible to build up a formal/semi-formal model based on informal sketching, and if it is possible to model and analyze alternative design concepts through a formal/semi-formal modeling language. Despite these proposals, there are still gaps between informal modeling used in the initial conceptual design phase and formal/semi-formal modeling, which are not properly addressed. Hence, assessing consequences (which is another target of the integrated design infrastructure) is still an unresolved issue, and hypothesis 3 remains unverified at the moment. Future work targets gaining further insights into this challenge in order to provide answers to research question 4. Future work also targets incorporating optimization techniques into the integration infrastructure to be able to optimize
design concepts whilst also assessing their consequences.
Bibliography


Qamar, Ahsan, Wikander, Jan, and During, Carl. Designing mechatronic systems: A model-integration approach. In 18th International Conference on Engineering Design (ICED11), To Appear, Copenhagen, Denmark, 2011b.


Suh, Nam P. Applications of axiomatic design to engineering collaborations and negotiation. In International Conference on Axiomatic Design (ICAD), 2006.


Torry-Smith, Jonas Morkeberg and Mortensen, Niels Henrik. A mechatronic case study highlighting the need for re-thinking the design approach. In *International Conference on Engineering Design (ICED11), To Appear*, Copenhagen, Denmark, 2011.


Appendix A

Designing Mechatronic Systems, a Model-Based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration

Ahsan Qamar, Carl During and Jan Wikander, Designing Mechatronic Systems, a Model-Based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration, in proceedings of the 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM09), Singapore, 2009

Copyright © 2009, IEEE. Reprinted with permission
Designing Mechatronic Systems, a Model-based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration

Ahsan Qamar, Carl During, and Jan Wikander

Abstract—Higher demands on efficiency, cost and functionality have contributed a great deal towards the advent of Mechatronic systems where mechanics, electronics and computer software integrate together to provide the required functionality. This integration has its effects in the design process as well, and a good design requires careful integration of methods and tools to satisfy the overlapping objectives. The set of requirements a designer needs to satisfy for a good design are in all three domains, hence various design and modelling tools are used by engineers to satisfy these multi-domain requirements. With the advent of systems modelling languages for specifying the complete system in one system model, there is an increased urge to link the system modelling tools to the domain specific tools such as Matlab/Simulink. In this paper we present an attempt to achieve an integrated design environment by building mapping between SysML and Matlab/Simulink models. The complexities that can be solved using SysML and the ones which necessitate a communication between SysML and other tools are discussed. Models of an industrial pattern generator are presented to explain the practical influence of this integration approach. The resulting integrated model is more comprehensive for the designer when investigating various design alternatives. This is important for companies considering they have to consistently embed innovation and sustainability in their products.

Index Terms—MBSE, SysML, Model Integration

I. INTRODUCTION

Innovating the products or services is important for any company. The process of product development starts with a novel idea captured as a perceived need. Transforming this need into a solution that can successfully satisfy the real need requires an appropriate design process [1]. An example design process (waterfall model) is show in figure-1. The perceived need for a new product/innovation is explored to ensure that the right challenge is addressed with due consideration to all the stakeholders. This creates an understanding of the real need which is very important as it is the basis for creating a well defined description of the intended design and of the design requirements [1]. Preliminary ideas are evaluated against the requirements giving design concepts which are then transformed into the final solution (product). The design requirements play a key role during the product development process as the final solution is traced back to the requirements to verify that it matches the required criteria.

Fig.1. A waterfall model of a design process [1]

The closed loop nature of whole process helps in going back and forth from requirements to concepts and then to solution phase (where functionality is designed) to ensure that the detailed design of the final product/service has characteristics as intended and perceived in the requirements. However effective follow up of this design process by the different participating design teams is a crucial factor in obtaining the final product. This was the major drawback of a document based approach due to inherent difficulties in maintaining the consistency and validity of the documents by the different design teams. The shortcomings of the document based approach are addressed by the concept of model based systems engineering (MBSE), where it is possible to maintain a good synchronization between system requirements and the evolving design (at various phases of the design process). Also variant system design options and the quality of the evolving design can be checked using the developed system model.

A. Complexity in Mechatronic System Design

The models developed to replace the documents provide great advantages as mentioned in [2]. However it is still important to understand the complexity that exists in designing a Mechatronic system, where the system engineering models represent multiple domain information which is used by the different engineering design teams. If we make a demarcation of the multi-domain aspects of the system in terms of the three aspects namely the Control

This work was supported in part by Micronic Laser System A/B and authors would like to pay gratitude to the company for providing the support in conducting this technical study.

Ahsan Qamar is with the division of Mechatronics at the department of Machine design, Royal Institute of Technology, Stockholm, Sweden (Phone: 0046-762127923; e-mail: ahsanq@md.kth.se).

Carl During is the director of technology development in the research and development department at Micronic Laser System A/B, Stockholm, Sweden, and also an adjunct professor in precision Mechatronics at the department of machine design, Royal Institute of Technology, Stockholm, Sweden. (e-mail: carl.during@micronic.se)

Jan Wikander is a professor in Mechatronics and head of department of machine design, Royal Institute of Technology, Stockholm, Sweden. (e-mail: jan@md.kth.se).

This is important for companies considering they have to consistently embed innovation and sustainability in their products.
Design, the Mechanical Design, and the Electronics Design (other aspects such as software design and electrical design are present but are omitted here for clarity). Then the systems engineering approach towards designing such a system will be through a system model containing a separate representation of the three aspects of the system. The interconnections that exist between the three aspects will also be modelled in the system model (shown by double sided arrows) as shown in figure-2.

As mentioned in [3], the problems encountered during the system development process are strongly related to the organization and management of complexity. This is also the main problem typically addressed in MBSE research. However, in advanced Mechatronics design also, the more concrete technological design integration and optimisation issues require multi-domain design methods and tools. Figure-2 shows the abstraction hierarchy related to the Mechatronic system design. The level of detail and preciseness changes while going from the system design level to the tool level. The system model helps in resolving the complexities at the upper abstraction level by transforming the multiple stakeholder objectives into simple and well organised information blocks (figure-2). However, the complexity at the domain specific model and the tool level still exists, reason of which can be interpreted through figure-2.

One of the reasons is that the intercommunication between the three aspects of the complete system is easier to capture in the system model, but the same is difficult to achieve at the tool level, where dedicated tools are used. This is also visible from figure-2 portraying no tool to tool communication (which then happens manually between team engineers). Referring to figure-1, the feedback nature of the product design process means that the design and simulation results be traced back to the stake holder needs. If there has to be a modification of goals suggested by a domain specific engineer, then it has to be verified if those modification still satisfy the stake holder objectives or not. This is impossible to achieve through models right now as no communication exists between the lower-level tools back to the system model.

In order to comply with the iterative nature of product design process in figure-1, the engineer at tool level manually communicates with the system engineering team to check if the modified objectives satisfy the multiple stake holder needs. In the case it does not, they have to go through the whole iteration process again. A similar problem appears when considering a change in the system model while comparing multiple designs for a specific component, hence creating a need of multiple design iterations to be performed at the tool level and between tool level and system level.

We argue that the complexity of multi-domain design and optimisation at the tool level is a major obstacle against a better product design process, and that current MBSE research does not properly address this problem. While considering a model based approach for Mechatronic systems, we want to answer questions like:

1. How to solve the complexity issues at the tool level
2. How the model information and parameters should be exchanged between different domains
3. How to address the communication from the tool level back to the system level
4. How the communication between the system model and the tools can be used to effectively overcome the need for tool to tool communication

This study focuses on exploring a system model (created using OMG System Modelling Language (SysML) [10]) and tool level communication possibilities and scenarios. We discuss about obtaining this communication using model level integration or model-tool integration. Here we specifically outline an integration process between a SysML model and Matlab/Simulink [11], with support from a case study which is being done on a semiconductor laser pattern generator (described in section 4). Essentially this study is our effort to answer the questions 1-4 we raised above.

Two examples are presented from the laser pattern generator (section 4) where it is shown how the SysML model integrates into the corresponding Simulink model, how the complete architectural model can be used to investigate the market stake holder needs, and how these needs can be tracked down to various Simulink models. The remaining part of the paper is structured as follows. Section 2 throws light on the SysML system model and how its abstration layers come close to the tool level. The scenarios and issues pertaining to integrating the SysML-Matlab/Simulink models are presented in section 3. Section 4 describes the semiconductor laser printing system from Micronic Laser Systems A/B, its main parts and functions. Section 5 presents the two examples from the case study, and aim at giving the reader a clear understanding of the integration process. The paper concludes by presenting a discussion on the expected outcomes from this study, along with a discussion on how a company like Micronic can use these for future design studies.

II. MODEL BASED DESIGN USING SYSML

To support model based systems engineering (MBSE), Object Management Group has recently developed the Systems Modelling Language (OMG SysML [10]). SysML is general purpose systems modelling language that enables
systems engineers to create and manage models of an engineering system using well defined visual constructs [10]. SysML extends the UML2 profile to support an application that extends beyond software engineering. The output is a system model (from now on we will refer to system model as the one obtained using SysML) containing system specification, design, analysis and verification information in four main SysML building blocks. These are structure, behaviour, parametrics and requirements [2]. One feature of SysML is the support for representation of continuous behaviour and flow rates. This feature can be explored for possibilities of simulating/executing the continuous behaviour (expressed in SysML) in a domain specific tool (such as Matlab/Simulink).

The system model includes components interconnections and interfaces, component interactions and related functions components must perform, and component performance and physical characteristics [2]. Hence the system model contains and generates information about component requirements for the component developer; the component developer can provide feedback about how the component design satisfies the requirements back to the system model through model data exchange. Hence SysML provides an opportunity to organise the system information from the requirements to the functional level, and maintaining traceability between functionality and requirements. The functional level is in turn the abstraction level which is the closest to the tool level (i.e. component or subsystem design level). This functional level can be explored for exchanging model information for simulation purposes or integrated design domain provisions. One of such example is presented in [12].

III. INTEGRATION ISSUES FOR SYSML-MATLAB/SIMULINK MODEL INTEGRATION

The complexity at the tool level requires communication between the tools, between the tool and the system model. We argue that the SysML system model can be adopted to eliminate the need for tool to tool communication. This will essentially be achieved by the tool to SysML model level integration. Figure-2 portrays that the information is represented at different level of preciseness and detail going form system model to tool level. We envision an integrated environment to solve the dependencies that exist between the domain specific tools, and between the components/functions satisfying the overlapping system requirements.

Scenarios requiring SysML-Simulink integration can be of different nature. The functionality can either be first perceived in Simulink, or in the system model. If it is perceived in Simulink, then it would be desirable to import it into the SysML model and vice versa [4]. We envision that a Simulink model could be constructed simultaneously as soon as there is enough information in the SysML model, or the functionality could be imported in the SysML model as soon as something is perceived in Simulink. This will require an exchange of model information and parameters between SysML and Simulink. SysML also provides provision for representing continuous behaviour (using mathematical equations). It is desirable to simulate and execute the continuous behaviour in Simulink. This would be possible by transforming one of the models to the other tool (model level integration). One such approach for Modelica language is proposed in [12]. The references [4], [5], [7] and [8] are examples of other efforts already done in this regard.

We will use the mapping between Simulink and SysML as outlined in [4], and take it a step further towards obtaining integration between a Simulink model of an industrial system and its system model described using SysML constructs. We will throw light on how the structural and behavioural mapping of various SysML and Simulink artefacts can be applied to an industrial Mechatronic system.

IV. SYSTEM DESCRIPTION

We wish to explain the model transformation procedure between SysML and Simulink by extending it to an industrial system application manufactured by our industrial partner Micronic Laser Systems A/B. The semiconductor high speed precision laser pattern generator from Micronic A/B is used for mass production of both semiconductor chips and displays. It is a complete Mechatronic system with a mechanical architecture, industrial electronics, automation and control. In terms of functionality, the system can be broken down to different functions such as crunch data (electronic image blue print), creating image, placing image (on the photomask), control and environment etc. In the pattern generator, the micro-lithography process takes place to image a pattern on a photomask. The photomask is in turn used by a stepper-tool to place the image onto a silicon wafer.

![Main components of the Precision XY-stage](image)

Fig.3. Main components of the Precision XY-stage [9]

The system is very complex as a whole, and we wish to focus only on the main stage functionality i.e. placing an
image onto the photomask. This is performed on an XY-table precisely controlled and monitored by a feedback system. The Place Image subsystem (XY-table) currently has a resolution (design nodes) of 45nm (smallest feature that can be printed on silicon). Figure-3 portrays the main components of the precision XY-Stage. Advanced optics is used to split the beam from a laser into multiple beams which are targeted towards the photomask in the lithography process. Both X and Y stage is actuated by high precision linear motors. The X and Y position of the photomask is monitored using interferometers that give a high precision measurement to the control system that controls the servo motors for controlling the X and Y position.

From the model based perspective, the control system for controlling the XY stage position was designed first using Matlab/Simulink. The system model of the stage is then made using SysML that specifies the structure (architecture of XY stage), behaviour model of various components, requirements and some continuous dynamics of the stage. It is desirable to import the Simulink model into the system model of the XY-stage to have a concrete picture of the stage behaviour and functionality in connection with its tool level realization. Both the models will be discussed further in section 5.

V. CASE STUDY

SysML uses nine types of diagrams to specify the system structure and the behaviour. It also provides support for specifying continuous dynamics (equations) and linking it to the underlying structural entities. SysML supports specifying the equations according to the modelling tool where they will be executed e.g. Simulink in this case. One main feature of a SysML model is that all the behaviour artefacts are allocated to structural components. Simulink on the other hand focus on behaviour only by specifying system dynamics in blocks executed over time. These dynamics are simulated to analyse if the desired goals are met or not. However seeing a system as whole, the simulation model in Simulink is only one aspect of the overall system description. It is important to describe how different parameters in the simulation model relate to parameters in other system description. Also it will be required to map the simulation results back to the system requirements. This is where transformation comes into play.

A. Mapping Simulink and SysML

We utilise the approach described in [4] to explain the mapping between various artefacts of both languages. A Simulink plug-in (developed in [4]) parses an mdl file (containing the desired Simulink model) and transforms it to the Eclipse UML2 framework according to the mapping design. The Eclipse UML2 model serves as the meta-model which is imported in the SysML system model. The Simulink model hierarchy is captured and the relationships between model entities are stored to transform the Simulink model in the UML2 framework (recognized by SysML) [13].

Blocks and lines are basic entities of a Simulink model. A block represents a system which might contain a subsystem. The subsystem is specified using Inport and Outport relationships. A line connects two blocks together. These modelling entities in the Simulink model are treated as objects in SysML, as both the languages are object oriented types. For example a block is represented by a block and a subsystem is represented by an internal block diagram structure. Signals between Simulink blocks correspond to SysML connectors with ports attached to it. Control flow and data flow through a Simulink connector can also be directly represented as control and data flow in SysML. SysML provides option for standard port requiring service based interface which is used in conjunction with flow ports to specify the Inport/Outport structure, and line representing flow/interaction between blocks in Simulink.

In terms of behaviour mapping, activity diagrams are used to represent the continuous-time and the sampled block behaviour in Simulink to its SysML representation. Stateflow in Simulink is represented by a State machine diagram in SysML. Simulink has explicit representation of time and clocks, where as SysML provides a non explicit representation. There is still a provision of specifying the timing by e.g. specifying continuous dynamics in SysML.

Now we focus on the XY-stage of the laser printing system and explain how the two models interact with each other. Placing the image onto the substrate is performed by the Place Image function which is a function represented in XY-stage system model. The complete system model of the Micronic laser pattern generator contains many components and functional parts described in various SysML structure and behaviour diagrams. Two of such diagrams are the Place Image function and the Level Base Stone sub-function. The following section presents the details.

B. Test Scenario -I

This example portrays a scenario where all different functions which constitute the Place Image function can be seen in the SysML internal block diagram in figure-4. It is important to understand that this diagram essentially represents how various parts of the laser printing system are interconnected to provide the Place Image functionality. These parts are described in other SysML structural or behaviour diagrams-details of which are omitted here.

For narrowing down our horizon, we will now try to put our emphasis on the Move Stage function which performs either of the stage X, Y, or Z motions when needed based on the input from the Adjust Image Position (for adjusting image onto the substrate) and Focus Image Control (for controlling the image focus onto the substrate) block property as shown in figure-4.

Since the Move Stage function consists of three sub functions (one for each X, Y, Z), we will only discuss here the Y-stage realization of it. It is important to observe here that it is the functional layer where the realization of different functions can be seen. One of such realization is shown in figure-5 where a Simulink model of the Y stage servo control is shown. This Simulink model will essentially be mapped inside the Move Stage Y block property using the
behaviour and structural mapping of the modelling entities as discussed above and mentioned in detail in [4]. The final result will be a SysML representation of the Simulink model (presented in figure-5) inside theMove Stage sub-function of the Place Image function. This will make the model complete in terms of functionality.

![Place Image Diagram](image)

Fig.4. SysML model of the place image function

This means that any changes that appear while investigating the various design scenarios/schemes can be seen in terms of its realization aspects as well (i.e. Simulink model). However, to simulate a new design realization, the Simulink model has to be built and simulated again using the new inputs that can be taken from its SysML counterpart. Hence this approach provides a complete overview of the functionality of a Mechatronic system/subsystem to the subsystem/component developer. We would like the new design scheme) automatically to provide to the designer.

One advantage of having a system model is that we can generate the component requirements (while investigating a new design scheme) automatically to provide to the subsystem/component developer. We would like the system model to be able to automatically generate the Simulink model itself, which would then be used by the engineers using Matlab/Simulink. This is a target we see as an intended goal of this study.

![Level Base-Stone Diagram](image)

Fig.6. SysML model of Level Base-Stone function

![Simulink Model Diagram](image)

Fig.7. Simulink model of the base stone velocity vibration control

C. Test Scenario - 2

This scenario presents a broader example following scenario-1 where we already described mapping of the Simulink model (figure-5) inside theMove Stage function of the SysML model (figure-4). From figure-4, we can notice another member function of the Place Image function named as Level Base Stone, which essentially means vibration control of the base stone in six degrees of freedom. The base stone is a solid foundation which holds the precision XY-stage on top of it, and any vibration of the base stone can propagate towards the XY stage. Such vibration is highly undesirable considering the extremely high precision demands. Therefore the base stone is provided with an active suspension to attenuate the vibrations, which is exactly what Level Base Stone function does.

Figure-6 shows the sub-functions of the Level Base Stone function which will be our focus now to present the second test scenario. A realization of Level Base Stone function is shown in figure-7, where the Simulink model of the velocity control loop of the base stone is shown. From our talk on how to transform the Simulink block entities to a SysML representation using [4]; this Simulink velocity control loop can be integrated into the two Level Base Stone sub-functions i.e. Create Levelling Servo Output and Move Base-stone with Levelling Actuators. What we achieve in the end is a SysML representation of the Simulink velocity control loop inside the Level Base Stone function.

The Simulink model of figure-7 is one of the three main control systems used in the laser pattern generator. The first one being the Y-stage servo control and third one is the X-stage servo control (which is almost similar to Y-stage servo control). Once we see the Place Image function in the system model and the Move Stage and the Level Base Stone functions containing their actual realizations i.e. the SysML representation of their Simulink models, we gain more information in studying a design modification, starting from the stake holder needs and down to the level where we see how the function is actually implemented.
As far as the control system design for the laser pattern generator is concerned, a mapping between the system model and these three Simulink models will complete the integration of control system design models with the system model, hence providing complete functional information from the system part as well as the realization part. For example a designer investigating a modification of linear motor for the precision Y-stage in the laser pattern generator will get a complete information about the change in system architecture through the SysML model, along with the change that will appear in the functional aspects (through the change appearing in SysML representation of the Simulink models of the three control systems). Complete information about the functionality of the laser pattern generator is thus available to even a system architect, and also to an engineer working with control system design in Matlab/Simulink. The reason being that studying a new design will show up changes down to the functional blocks in the system model (where Simulink representation is mapped), which serves as a base for the new representation of it in Simulink.

VI. RESULTS AND DISCUSSION

The designed functionality of the Mechatronic system is realised at the tool level in the domain specific tools. Technological growth in these tools to bring innovative ideas into reality means that designing everything in a single tool is unrealistic. Today the dynamic analysis and mechanical design tools give a great advantage of simulating the intended design under every possible real world scenario that it will encounter. We wish to keep the in depth analysis capability of each tool and propose a communication framework between the model based system engineering efforts and the domain specific tools. We argue that this approach not only utilises the domain specific advantages of each tool, but also strengthens the design environment more by forming an integrated design framework, where each participating tool teams up with the system model. It will provide a powerful and much needed design environment for Mechatronic system design, with bringing solutions to handling complexities in satisfying the challenging design requirements.

Engineer and designers perceive a new design either as a new functionality or a new system. They realise their vision, by modelling the new functionality in the domain specific tools and a new system is modelled in the system model. We argue that the integrated design approach for designing the multi-dimensional functionality should foster from the bottom where functionality is realised i.e. the tool layer and then move upwards reaching the system model and ultimately catching the stake holder demands. The resulting approach will be innovative and more open to accept bright ideas for product development, where innovative minds directly perceive their design in the corresponding domain specific tool and still managing to integrate it in terms of a complete system through a system model.

Through this paper, we have highlighted how the SysML system model simplifies the complexities in putting the multiple stake holder objectives into complete system design information. We have also explained how the complexities still appear down at the tool level, where currently a system engineer manually communicates with the engineers working with these tools, and try informing them about the objectives in each design domain. Indentifying the problem is a main step towards finding its solution. Our discussion about these tool level complexities has not yet been attacked much by researchers, which hinders in following the feedback process mentioned in figure-1.

This paper has highlighted how a Simulink model can be mapped into the corresponding SysML block. One of the two demonstrated examples was of the Place Image function for the laser printing system, which was modelled first in Simulink, where a control system design for Y-stage was simulated. Later this model was transformed and mapped inside the Move Stage block, a function modelled in SysML model of Place Image function. We have explained what issues appear in the mapping process and how the two models communicate with each other. Using the example of this industrial system in both SysML and Simulink, we explained that the functional layer of the system model could be customised to import a Simulink realisation into it. It is important to emphasize that the opposite could also happen, i.e. the function is modelled first in SysML, which is required to be realised in Simulink. In this case the transformation will be opposite.

REFERENCES

Appendix B

Integrating Multi-Domain Models for the Design and Development of Mechatronic Systems

Integrating Multi-Domain Models for the Design and Development of Mechatronic Systems

Ahsan Qamar, Martin Törngren, Jan Wikander
Department of Machine Design
KTH, 100 44 Stockholm, Sweden
{ahsan.qamar, martin, jan}@md.kth.se

Carl During
Micronic Laser Systems AB,
Taby, Stockholm, Sweden
carl.during@micronic.se

Abstract. Design of mechatronic systems is driven by engineering efforts in multiple-domains, resulting in models developed in various formalisms. In spite of interconnections between these domains, approaches linking them theoretically are lacking. This lack of integration leads to major challenges in solving dependencies across different domains, slowing down the design process. In this paper, a model-level integration-framework is described, aiming to identify and solve dependencies across different domains during the design process of a mechatronic system. A two-degree of freedom robot example is presented, to exemplify the iterative process of design optimisation in a computer aided design tool (Solid Edge), connected with corresponding dynamic analysis and controller design in Matlab/Simulink. A system model built in systems modelling language (SysML) is extended to capture the cross-domain dependencies as parameters in a black-box model. These black-box models can be systematically linked with other domains (CAD and dynamic analysis) through the model integration algorithm (based on the robot example). As the modellers perform the design iterations in Matlab/Simulink and Solid Edge, the updates on cross-domain dependencies will be available through SysML model, containing the complete system view. Possible issues with un-identified dependencies (due to change in requirements), and their possible solutions are also discussed. It is emphasized that the robot example could be generalized towards mechatronic systems, along with the requirement specifications for the model-integration algorithm. It is demonstrated that efficient design solutions and reduction in design time are possible with concurrent multi-domain models integration.

Introduction

Background. Massive pressure on introducing the state of the art functionalities in products and services along with constraints on time, cost, and quality creates high competition between companies. To deliver reliable products and services at rapid rate, improved design approaches are adopted. Maintaining the quality and cost-efficiency of the products while developing them at a high pace clearly constitutes an important industrial challenge. An example is mobile phones, where a new mobile comes to the market almost every single month. Although the basic structure of these products remains the same, however, various changes are introduced in terms of functionality. Therefore, companies targeted technologies in which they can directly transform a new user or market need to the functional specifications of the product, which are then transformed to a product function. Balancing new functions with non-functional constraints such as energy consumption and usability during the product development process is also a key challenge. Minimizing the development time from identifying the need to providing the solution inside a product is a key for survival and becoming industry leaders. Models that contain user needs, design specifications, analysis and simulation information, and implementation algorithms are all key part of the development process. An important consideration is how various models can integrate with each other, for the product to function
without errors. Solving these integration requirements is a major challenge for various engineering design teams working on different product aspects in an industry.

Mechatronic systems are aimed to be highly modular, decentralized, and well integrated at the same time during the design phase. While mechatronic systems indeed incorporate parts constructed by different engineering disciplines, the actual co-operation during the construction is less developed. Although there have been substantial industrial and research efforts, integrated development processes, tool-chains, modelling formalisms, and joint analysis are still lacking. Every discipline has its own approach, and an integrated framework for the construction of mechatronic systems is missing, Wikander et al. (2001), Schäfer (2007). The guidelines presented in VDI2206 (1993) attempts to bring together the domain specific methods, and promotes inter-disciplinary communication for successful mechatronic system design. However, software engineering methods still have to evolve to support the design/modelling of these multi-domain systems. This evolution is clearly towards building bridges across different modelling formalisms, with incorporating model transformation themselves inside different models to build these bridges.

The complexity in designing mechatronic systems exists both at the top level; where requirements have to be specified formally for the complete system, its subsystems, its components and its behaviour, and at the bottom level; where simulations and analysis and in-depth product design is performed (Qamar 2009). Many of the hidden complexities in the system become visible going from one abstraction level to another (Holt 2008). Creating models at different abstraction levels is necessary to unleash the hidden complexity present inside the system which is otherwise not visible while modelling at a single abstraction level. This is a major reason that to handle the complexity in designing mechatronic systems, many researchers have developed modelling solutions that carry the advantages of both systems engineering at a higher abstraction level, and the detailed design tools at a lower abstraction level.

**Model based development.** Systematic development of products and systems is based on a development process. The development process ensures that the requirements are captured based on the specified stakeholder and market needs, which forms the foundation of the system to be developed. Hence, everything from the system specifications, design ideas, to the final design implementation depends on these requirements. Following a model based design approach; all those steps are systematically performed in different models developed in various tools. Typically for mechatronic systems, the V-model (Forsberg 2000, 1992), VDI2206 (1993)) development process is used as shown in Figure 1. The V-model adapts to the different design abstraction levels, starting from a higher abstraction level capturing the stake holder requirements to the lower abstraction level where the detailed design is analyzed and implemented. Following the development of model based design tools over the past decade to facilitate the product design process, it is not possible to have a single tool following the complete V-model and giving views at all levels of abstraction. Hence different tools are used by engineers in the development of mechatronic systems. However, a comprehensive modelling approach requires sharing of information between different tools, to achieve an integrated design and development environment.

In this study we will place emphasis on three different design domains typically involved in the development of mechatronic systems. The first domain is the *systems engineering approach* towards the physical system design, which has been addressed by the Systems Modelling Language (Systems Modelling Language Specification 2008) from the Object Management Group (OMG). SysML supports capturing the requirements of a system and creating structural and behaviour models. Moreover SysML has a capacity to represent the equation based
behaviour through constraint objects, for example to represent continuous dynamics of the system. On the other hand, tools such as Matlab/Simulink (Mathworks 2009) and Modelica (Modelica 2008) are widely used for analysing the continuous dynamic behaviour of the physical system, along with the control system design which is part of the second domain. The third domain is the computer aided mechanical and electrical system design. Both the second and third domain are used for the detailed system design in connection with the system model from the system engineering domain as shown in Figure 1.

**Figure 1: The V-model development process, VDI2206 (1993), Forsberg (2000, 1992)**

Creating a complete system view from the higher abstraction level to the detailed analysis level is important for the future product development, where the models can be reused in a new design scenario. The information model developed using SysML shall contain all the changes and modifications that are made to the initial design and it will continue to be further developed as the product is revised in its life cycle. Hence the connection of the detailed-design models with the information model is very important, for reusability of the developed models.

**Complexity in mechatronic system-design.** Recently systems engineering model created using SysML has been studied by scores to represent multi-domain aspects of mechatronic systems. Figure 2 highlights the complexity resulting from modelling multi-domain aspects at different abstraction levels. The multi-domain aspects and their interconnections can be easily modelled in SysML shown by double sided arrows in Figure 2. The modelling aspects differ by changing the abstraction level, i.e. the system model contains a complete picture of the system at a higher abstraction level. The detail design is however done at lower abstraction level (tool level) in different tools. By changing the abstraction level, we bring out two inherent complexities. The SysML model (green-box in Figure 2) helps in resolving the first level of complexities by transforming the multiple stakeholder objectives into simple and well organized information blocks (Figure 2). However, the second level of complexity (blue boxes in Figure 2) still pertains, where different tools are used to perform simulation and analysis, and these tools have to explicitly capture the sub-system design requirements broken down from the stake holder requirements. The models in these tools also perform design iterations on parameters which cannot be optimised stand alone, rather are dependent on models in other tools. This can be understood by comprehending Figure 2, where interconnections and dependencies between control, mechanical and electrical is possible to be modelled in SysML model (shown by double sided arrows), however the same is difficult to achieve at the tool level. Hence Figure 2 does not show any tool to tool communication.
Evaluating parallel/competing concepts against the design benchmarks during the development process requires translating those concepts to a detail design tool for simulation and analysis. Another important issue is connecting mechatronic system design V-model (Forsberg (2000, 1992), VDI2206 (1993)) with the modelling topology. The feedback nature of the V-model design process means that the simulation and analysis performed to reach the final iterated design, needs to be mapped back to the system and sub-system requirements, mentioned in the SysML model. While referring to requirements, it is important to understand that the requirements cannot be regarded as fixed but are evolving during the project life time, either due to a market need, or from a change initiated by a modeller working in a detailed design tool. This requires an integrated design methodology, which not only helps in solving dependencies between models, but also helps to analyse the design alternatives from both a top level (system level) to the detailed design level. Integration will also help in keeping track of evolution of requirements along with evolution of design, and to trace the final result back to the requirements in the system model.

**Figure 2: The complexity in the system engineering model including the domain specific models and tools (Qamar 2009)**

**Related Research**

SysML is a general purpose language aimed to specify systems as concretely as possible; however it stands together with other detailed modelling and simulation tools. Using SysML, a modeller can abstract a domain specific language to a level that permits its interaction with other system models (Paredis, 2008). Hence, several researchers have focussed towards achieving an environment where SysML capability of precisely defining and specifying the physical system in a graphical information model could be combined with the design, analysis and simulation models in other tools. Many such efforts have focussed on integrating SysML with Modelica (Modelica 2008) and Matlab/Simulink (Mathworks 2009). Integration with CAD tools appears to be studied to a lesser extent, though there do exists CAD tools that connect to a Matlab/Simulink environment for further dynamic analysis and controller synthesis. However, dynamic analysis of the system, its control system design and electrical design is usually performed assuming that the mechanical system will not change. While system designers in an industry visualize a new design in terms of the new performance and function benchmarks, it is not regarded necessary to know the details of CAD models forgetting that the CAD is done concurrently. It can be argued that a better design performance could be achieved being able to optimise the mechanical design simultaneously with other optimisation procedures. At the same time, it is emphasized to follow a top-down design approach (the V-model) towards the product design. Current research on communication between detailed modelling tools such as between CAD tools and Matlab/Simulink only
provides an integration capability at a lower abstraction level. For a broader system view, analysing different design alternatives in detail, and for provision of the optimum design concepts inside the products at a rapid pace, integration of system model with the detailed design tools is crucial. This is due to the fact that the realization of the new functionalities from a system perspective (connecting to the requirements) to the analysis and implementation perspectives is important for different set of modellers involved in the product design. A top down approach has the capacity to provide such solutions.

**Model-level integration.** Executable UML (Mellor and Balcer 2002) is an example of adding more details to a language for creating executable models, by specifying system using a UML (Unified Modelling Language 2009) profile and generating testable and compilable models through them. SysML being based on a UML2 profile has the same capability i.e. by adding more details to a SysML model to generate executable models. Huang et al. (2007) explored the potential for SysML as a modelling tool to support simulation by creating design and analysis models in SysML and mapping them to executable models. Jobe et al. (2008) classified the system in different analysis objectives (based on modeller targets) modelled as aspects in a SysML model, to achieve model reuse, which allows deploying the semantics of one design alternative to another in a same design problem, or to another design problem within the same application. Brisolara et al. (2007) investigated mapping of UML models to Simulink through the ATL (Atlas Transformation Language, 2009) transformation engine reading a UML model specified in an Eclipse framework (Eclipse Modelling framework (EMF) 2009) compatible tool. This mapping allowed one to exploit the benefits of UML while obtaining the executable code for multi-processor architecture through the generated Simulink model. Mapping Simulink and UML models is also investigated by Jianlin Shi et al. (2007) and Sjöstedt et al. (2008a) for the design of embedded systems in particular.

Many researchers have targeted integrating continuous dynamic Modelica (Modelica 2008) models with SysML/UML models. The UMLH profile was developed by Geusen (2007) to specify hybrid systems Modelica models in UML/SysML, and generating Modelica code from them. Pop et al. (2007) investigated integration of Modelica and UML through ModelicaML, a SysML profile. The ModelicaML reuses many SysML/UML constructs alongside the constructs like the equation diagram (for specifying Modelica equations) and the simulation diagram (for specifying simulation experiment information). The simulation diagram is integrated with the Modelica solver to run a simulation experiment. The work by Schamai et al. (2009) contributed further to the development of the ModelicaML profile where the authors also provide details about how textual requirements can be specified in a ModelicaML model, specifying a behaviour for a requirement which can then be simulated using the Modelica solver to deduce if a requirement is satisfied or violated. This is a promising notion to ensure traceability. The authors also provide details about validating ModelicaML models and simulating and enhancing the simulation environment using OpenModelica (Fritzson 2007).

ModelicaML and UMLH are based on creating a separate UML profile with the new language constructs created explicitly for the target modelling tool. However Johnson et al. (2007, 2008) and Paredis et al. (2008) relied on SysML’s ability to integrate models by using the available SysML constructs and by extending the semantics of SysML rather than creating a new profile. The authors have investigated how to integrate models of continuous dynamics into SysML by adding reusable modelling elements to the SysML model that are stereotyped explicitly for Modelica language. Correspondence rules were written between SysML (using stereotypes) and Modelica language, and an automated model transformation approach using Triple Graph Grammar (Schurr 1994) has been shown. Similarly to this work, Sjöstedt et al. (2008b) showed limitations with SysML parametric diagrams for modeling acausal dynamic systems and possible ways to overcome this problem.
Research need. Integration is one of main problems existing in designing mechatronic systems in general. This study focuses on utilising the SysML information model especially the parametric models along with extending the SysML semantics to integrate with models of continuous dynamics, and CAD models especially. For mechatronic systems, which are optimized against multiple objectives in different design domains; this approach will connect state of the art optimisation methodologies in different domains. The SysML meta-model is extended to create customized semantics purposely created for integration support for Matlab/Simulink (Mathworks 2009), and CAD tools (Solid Edge 2009). The studies done in (Johnson, 2007), (Johnson, 2008), and (Schamai, 2009) have not considered mechanical design through CAD models, the reason being that modifying the mechanical structure is seldom considered in a traditional design approach. However, if the models are integrated to solve the dependencies between them, then modification of a mechanical design in connection with its dynamic analysis and control system design could bring much better optimisation results. Two degrees of freedom (DOF) robot example is explained. Different modelling formalisms for this example are analyzed to deduce conclusions on what kind of requirements are present for solving dependencies for a physical system. This case study is explained in detail in following section.

Case study

The two-DOF robot example aims to explain the design processes for a mechatronic system. The motivation behind using this robot as an example system is because its solution in CAD and the corresponding dynamic analysis is easy to develop. Thus the complexity faced during modelling and design optimisation of the two-DOF robot system, mainly resulted from the dependencies between different models. These dependencies exist only due to lack of integration, and not due to the limitations of the modelling tool. A few questions that need to be addressed through this example are:

- Finding dependencies and overlapping inter-connections that exist between models created in different tools.
- Explaining the iterative process from the start to reaching the final design, while refining and tuning various parameters, either local to a certain model, or shared between different models.
- Defining how the iteration process in one modelling formalism affects design parameters in another model, due to dependencies across different domains.
- Categorizing and presenting the information generated as a result of cross-domain-dependencies inside the SysML model, which is used as a common knowledge-base.
- Knowing the shared and local parameters, how to formulate the general requirements for system integration based on the two-DOF robot example (the underlying goal is to further extend the robot example towards general mechatronic system design problems).

Answers to above questions will contribute towards the development of an intelligent model information exchange algorithm, based on formulating which information a modelling formalism requires from the other, and how to create the bridges to provide that information. Two-DOF robot example provides sufficient analyses to answer such questions. Figure 3 illustrates the envisioned design setting.

Based on the block representation in Figure 3, the following sections will present three different modelling formalisms: using SysML as a common system modelling formalism, along with the mechanical design in Solid Edge, and continuous dynamic behavioural analysis and controller design in Matlab/Simulink. Each model will be analysed from the point of
beginning of modelling effort to the end. The dependencies that arise while modelling across various domains will be highlighted. A solution to these dependencies based on knowledge gained from the robot example will be discussed.

**Pen & paper information model**

Figure 4 portrays the start point of the modelling process for the two degrees of freedom robot system. This type of initial model comes into shape by brainstorming on a paper or a drawing board, from different designers/engineers. Hence it is called *pen & paper model*, capturing certain design information. It should be kept in mind that for complex systems, it might not be possible to draw such a pen & paper diagram (as in Figure 4) containing even the structural geometry; however a rough system and sub-system sketch is usually possible along with the textual information present in the pen paper model specifying the initial design goals.

![Figure 3: Two degree of freedom robot modelling example in three domains, model integration algorithm behind the model information exchange layer](image)

**Figure 3**

**Pen & paper information model**

Figure 4 contains information in terms of parameters (e.g. LA), and design context (e.g. robotic arm containing two revolute joints). Typically, the design of a new product flourishes through such captured information sets, which designers capture inside a pen & paper model. We aim to design two degrees of freedom robot satisfying a planar work-space requirement. Different properties such as: robot geometry parameters (containing the structural information), range of rotation of its joints, and placement of robot inside the workspace, all come together to define the work-space of the robot. Hence, all available information inside Figure 4 can be classified as design specifications that come to life following a model based development framework. Applying the same idea to Figure 4, a list of properties can be obtained, which will be designed through the detailed modelling in a CAD tool (Solid Edge) and Matlab/Simulink, as mentioned in the following:

1. \( L_A, L_B \) (link length’s), \( W_A, W_B \) (link width’s), \( \theta_A, \theta_B \) (angular position of each joint)
2. Pivot point for each arm: Placement of point of rotation on each arm

![Figure 4: Pen & paper model for two-DOF robot](image)
3. Origin point for first arm in the workspace
4. Joint space/range and joint type: The position of each joint at certain time instant \((\theta_A, \theta_B)\) is constrained by the range of movement of each joint
5. Workspace: Link length’s together with the position of each joint defines the robot workspace

**System modelling using SysML**

Model based concurrent system design solicits that modeller working in each tool requires the necessary information about the system (requirements, contexts, parameters etc) to start with the modelling process. Using a top-down framework, it is desired for the SysML model to contain the necessary information that a CAD tool and Matlab/Simulink require, for the corresponding analysis to begin (common knowledge-base). It is also possible that a rough CAD model and a Simulink model are created before starting with the system modelling in SysML. This usually happens with no previous system models available to be reused (bottom-up design). Nevertheless, industries require/employ parallel design efforts in each of the concerning tool, to minimize the development time, and modeller in detailed analysis tool could have a rough visualization before they receive the system specifications through a SysML model (ideally).

![Figure 5: Block definition diagram (bdd) for two-DOF robot, specifying different sub-systems and their components](image)

A SysML model of two-DOF robot system has been created, containing information about the complete system in terms of its structure, its behaviour, its requirements, and constraint based behaviour using parametrics. We will present a few excerpts from this SysML model, to help visualise how the SysML model has the capacity to be extended and serve as a common knowledge-base not only for streamlining the modelling activities in the detailed modelling tools, but also to capture the cross-domain-dependencies inside it. Figure 5 presents the two-DOF robot structure along with design requirements.
Deductions. In order to answer the questions (presented earlier in this section), the knowledge gained from building-up a SysML model is critical. Some important deductions are discussed below:

- The part/value properties such as link_length of the component link contain the geometrical data, which is designed and iterated through the Solid Edge model. Various properties can be connected using defined constraints to generate a derived property, for satisfying a design requirement.

- A parametric diagram connecting properties such as link_length and JointAngle (shown in Figure 7) represents the forward kinematic model of robot. The constraints e.g. Linktransformation_zrotation (mentioned in Figure 6) are written in Matlab syntax, the intended execution language.

- It is desirable to execute the robot kinematic model in Simulink. Either the parametric diagram could be transformed (Johnson 2008, 2009), and the result is traced back using a bidirectional transformation (white-box approach), or the input/output parameters connected to the parametric diagram could be communicated to Simulink, where the kinematic modelling is done (black-box approach).

- A physical parameter such as link_length_A could be modified in Simulink, which has to receive the consent of the CAD designer. These dependencies can be communicated through the SysML model, where an update on these parameters will be available. Thus the impact of a change in one view translates to the other view using SysML as a common base. While design iteration in Matlab/Simulink will aim at optimizing the system dynamics and designing the controller, the dependencies between Matlab/Simulink and CAD will bring a change in the CAD model too and vice versa. The whole integration workflow is available in the SysML system model, which is critical to connect different views of the system.

- Parametric diagrams can be built for different contexts. The input parameters and output parameters are iterated between CAD and Simulink, and any modification will go through the SysML model. The remaining blocks that connect those parameters will be inside Simulink or CAD model. A final target will be to represent some explicit blocks (e.g. controller transfer function) inside the SysML model, which is a sub-set of the white-box approach.
We have emphasized that iterating a mechanical design along with its dynamic analysis and controller design leads towards better optimization of the complete system. Hence, the effects of changing a design parameter need to be communicated between all models which utilise that specific parameter. This communication will be the backbone of the solution to handle cross-domain dependencies. The mechanical design will be based on requirements from SysML model on the robot workspace. Thus it is important to pass the relevant requirements to the Solid Edge model, which in this case is workspace requirement as shown in Figure 5. Analysing the steps while creating the model of the two-DOF robot in Solid Edge, a block representation of the procedure can be created as shown in Figure 8. Figure 8 gives an indication of requirements between a Solid Edge model, SysML model, and Matlab/Simulink model.

In the context of the two-DOF robot example, the power requirements for each motor and sensor precision requirements are considerations that depend upon control system objectives as well. Based on the initial workspace requirements, the mechanical designer can select the approximate motors and sensors, however they cannot be finalised without the dynamic analysis and controller design (see Figure 8). There is another input named change physical parameter. In the context of this example, the engineer working on the control system design in Matlab/Simulink can recommend changing a physical parameter (e.g. L_A) based on the kinematic analysis of model in conjunction with the design of a controller. Hence this is another dependency to be considered for optimising the physical structure. The same goes for the selection of motors and sensors that have effects on the mechanical design.

Based on all the necessary considerations, the mechanical design is finalised and the corresponding workspace of the design is analysed thus generating the geometry information for the two-DOF robot as shown in Figure 8. This information will be used within the SysML model to build information about the mechanical structure along with the Matlab/Simulink to carry out the subsequent analysis and simulation on this structure. Figure 8 shows a feedback loop type structure from the Solid Edge model to the SysML model and the Matlab/Simulink counter parts. This feedback loop identifies requirements between various models, which are the foundation of an integrated modelling framework.
Deductions. Some important deductions can be made from the Solid Edge model in the scope of model integration

- Relevant requirements described inside a SysML model are needed to begin modelling in Solid Edge
- Developing a Solid Edge model concurrently with SysML and Matlab/Simulink, it is possible to identify the parameters that the Solid Edge model requires and the parameters that the Solid Edge model generates. Hence we can conclude the explicit requirements for each model and for the integrated modelling approach as a whole.
- A better optimisation of mechanical structure is possible based on creating active linkages with SysML and Matlab/Simulink
- Most of the generated information could be captured as parameters, e.g. $M_A$ for motor-A specification, containing power of motor A.
- In an industrial development, very soon the pen and paper model becomes a rough CAD drawing to visualize the system where parts can be manually moved along the constraint geometry.

Modelling continuous dynamic behaviour in Matlab/Simulink

Albeit it is possible to carry out the dynamic workspace analysis of the two-DOF robot in current CAD tools, the mechanical design cannot be finalised without a dynamic analysis and a possible controller design in a specialized tool such as Matlab/Simulink. The kinematic analysis of the robot is already presented as a parametric diagram in SysML, which needs to be simulated. It could be possible to transform the parametric diagram to a Matlab/Simulink model, where it is executed. However, we argue that in the long run, it is better to use the kinematic model as a black-box model in SysML. This means that only the input parameters to the kinematic model and the resulting output will be known inside SysML. The rest will be modelled inside the relevant detailed design tool (Matlab/Simulink). The reason being that kinematic analysis is more suitable to be performed in Matlab/Simulink than in SysML. What is rather important are the requirements and information that are needed to build the same kinematic model in Matlab/Simulink. The kinematic model is further used for designing a control system for the robot as shown by red-box in Figure 9.
The geometry information as concluded from the Solid edge model is the starting point for modelling in Matlab/Simulink along with the controller design objectives obtained from SysML model (shown by orange arrows). These controller objectives are based on the position control requirements as mentioned in Figure 5. The simulation and analysis in Matlab/Simulink will result in: finalising the power requirements for each motor based on optimising the controller, recommendation on sensor precision based on position control requirements, and a possible change in geometry properties of the two-DOF robot e.g. link length of Arm-A ($L_A$). The green arrows show the subsequent connection of parameters between Matlab/Simulink, SysML and Solid Edge. For the completeness of information in the SysML model, it is required that the final iterated controller model to be also parsed to the SysML model along with the kinematic model (built in Simulink). This represents the white-box approach, which will be a future goal of this study.

**Deductions.** Following knowledge can be gained from modelling in Matlab/Simulink.

- Parsing relevant requirements from SysML model to the Matlab/Simulink is paramount
- The requirements can be drawn for modelling a system across Matlab/Simulink, Solid Edge and SysML, solving cross-domain dependencies
- A similar feedback loop structure can be observed to be existing while modelling in Matlab/Simulink, the very essence of an integrated modelling approach

---

**Figure 9: Modelling process for creating the Matlab/Simulink model of the two-DOF robot**

The process of developing multi-domain models and attacking the complexity in designing a mechatronic system has been explained. Using a common knowledge base (as SysML) provides a comprehensive view into the system design. At the same time, the dependencies across different domains can be identified and managed. This paper discusses how to solve these dependencies by building bridges between SysML and other detailed-design models, and parsing different parameters over those bridges. The algorithms (*model information exchange layer*) that will make possible sharing those parameters are under development. More importantly, it has been identified what needs to be communicated between different tools for the two-DOF robot problem. This could be generalized towards mechatronic systems, where a similar pattern with designs in CAD and Matlab/Simulink is observed. Hence, this approach

---

- $L_A$, $L_B$
- $W_A$, $W_B$
- Pivot point
- Origin of arm-A
- Angle-space for each joint
- Motor and Sensor spec
- $\theta_A$, $\theta_B$
- Forward Kinematic model predicting Robot End Position
- Controller to avoid obstacle
- Power requirements for motors
- Required sensor precision
- Change physical parameter

Matlab/Simulink
carries significant potential to provide necessary infrastructure in terms of design tools, and handling the market pressure for rapid design of products.

**Extending SysML.** SysML profile can be extended to support information flow for both CAD and Simulink. UML profiling approach is used to create two new stereo-types- Sim_Simulink, and Sim_Solid Edge (see Figure 10), which are explicitly developed to contain input/output parameter information for simulation and analysis in the pertaining tools. Apart from the input and output parameters, rest of the model exists either inside Matlab/Simulink or inside Solid Edge (black-box model). The black-box approach provides advantage of starting concurrent design in detailed-design tools based on information obtained from input/output parameters. After iterating and reaching a final design, black-box models can be converted to white-box models. For example, a control loop with a controller transfer function designed in Matlab can be later represented inside SysML model, showing its connection to input/output parameters. This transformation from black-box model to white-box model is future work of this study.

The input/output parameters represented inside Sim_Simulink and Sim_Solid Edge contains dependencies that exist because both models share the same parameter in optimizing a design solution. For example, link length for arm-A \(L_A\) will be designed in Solid Edge. This can be changed in Simulink to a new value, however, since the black-box model in SysML involve \(L_A\), the change about \(L_A\) will appear in Sim_Simulink. Hence, the same change will appear in Sim_Solid Edge which comprise \(L_A\) as an input parameter, consequently communicating it to the Solid Edge modeller. If Solid Edge modeler changes \(L_A\) again, this change will journey in the same way towards Matlab/Simulink modeller. Therefore, systematic model-level integration could be attained by sharing those parameters, which essentially yield the requirements to accomplish integration between SysML, Solid Edge, and Matlab/Simulink.

![Figure 10: Black-box model in SysML portraying the input/output parameters, which are used for detailed-design in Simulink and Solid Edge](image)

**Further Work.** It is vital to contemplate that the listed parameters in the black-box approach are for the components or functions, which are already envisioned from requirements to the
structural and behaviour description in SysML. If something is not captured in the requirements and identified later by modeller working in detailed-design tools, then this will create new parameters. An example is if the CAD modeller adds a new motor to the two-DOF robot for an extra functionality. However, this motor does not exist in SysML model; therefore, none of the parameters in the Sim_Solid Edge black-box diagram are related to the new motor. This scenario typically exists for a bottom-up design approach. However, it is possible to use model transformation algorithms to generate a new Motor-C block when transforming from Simulink model to a SysML model, to solve such issues. Similar approach is utilised for transforming controller transfer function and kinematic model (developed in Matlab/Simulink), to represent them inside SysML.

The proposed framework requires higher modelling efforts than a traditional approach, especially while identifying and modelling dependencies. At the same time, ensuring consistency among developing models through the integration framework is a challenge in itself. However, this framework has capacity to be developed and utilised as a backbone for the provision of a comprehensive design environment. Hence, the modelling overhead is negligible when compared with immense advantages of solving dependencies and achieving integration between various models. Although the two-DOF robot example does not represent complex systems present today, however, it provides enough complexity to analyse the underlying dependencies, modelling processes, and modelling problems in various domains. Therefore, a verification of proposed integration framework through the robot example is vital for its further generalization towards design of mechatronic system.

Conclusion

This paper highlights the complications in designing systems of multi-domain nature. Different types of interaction take place between modules of a mechatronic system. It is critical to capture these interactions inside models during the design of those modules. This paper presents two types of complications: one about identifying and representing system dependencies inside a model, and other regarding establishing connections between detailed-design tools and system engineering tools, to obtain a broader perspective into the system design. Two possible solutions have been presented, one by extending SysML model containing an explicit representation of dependencies. The other through model information exchange layer which builds bridges between SysML and detailed-design tools. For the robot example, three modelling formalisms (SysML, Simulink, and Solid Edge) are presented from the integration perspective, and in a way that they can be generalized towards common mechatronic systems. Main aim of this study is to define the requirements for a novel algorithm, integrating a set of modelling tools. Scenarios such as design alternative comparison could also be handled by communicating the model context (as input parameter) to the detailed-design tool, and coming up with a set of solutions for each alternative (as output parameter). These solutions can then be compared against requirements in SysML model. The proposed integration approach utilising parameter sharing through black-box models is a cornerstone in yielding an integrated design methodology. We argue that this approach has differences from the state of the art; however, it carries advantages for integrating CAD models, and iterating the mechanical design along with system dynamics and controller design. Hence, this approach may implement a necessary breakthrough towards physical system design.

References


http://www.eclipse.org/modeling/emf/


Schurr, A. Specification of Graph translators with Triple Graph Grammars (TGG). WG’94 Workshop on Graph-Theoretic Concepts in Computer Science. 1994.


Siemens PLM Software. Solid Edge.


**BIOGRAPHY**

Ahsan Qamar is a graduate student at the Division of Mechatronics at the Royal Institute of Technology, Stockholm, Sweden, where he is working towards his PhD. degree in Mechatronics. Having a background in automatic control, robotics and embedded systems, his research is focussed on study of multi-disciplinary aspects to support the design of mechatronic systems, design processes and tool integration for multi-domain systems, and system design in the scope of model-based development.

Martin Törngren is a Professor in Embedded Control Systems at the School of Industrial Engineering and Management at Royal Institute of Technology (KTH), Stockholm, Sweden. He is also a director of Innovative Centre for Embedded Systems (ICES) at KTH, and member of the KTH faculty board. His research interests are in distributed control systems, model-based development, model integration and management, architectures for embedded systems and design methodology. His research addresses various system aspects such as safety, reliability, performance, cost and flexibility.
Jan Wikander is a Professor in Mechatronics at the School of Industrial Engineering and Management at Royal Institute of Technology, Stockholm, Sweden. His main research interests are integrated mechatronics design in general, design and control strategies for mechatronic, robotic and haptic devices. Main fields of application are automotive, automation and medical devices. 12 PhDs have graduated under his supervision; he holds three patents and has co-founded two spin-off companies.

Carl During is a director of technology development at Micronic Laser Systems AB, Stockholm, Sweden, and also an Adjunct Professor in Precision Mechatronics at the department of Machine Design, Royal Institute of Technology, Stockholm, Sweden. His research interests are in precision stages for laser pattern generators, control, sensors & actuator technology and utilizing system engineering together with model based engineering in the development process.
Appendix C

A Mechatronic Design Infrastructure Integrating Heterogeneous Models


Copyright © 2011, IEEE. Reprinted with permission
A Mechatronic Design Infrastructure Integrating Heterogeneous Models

Ahsan Qamar, Jan Wikander
Department of Machine Design
School of Industrial Engineering and Management
KTH Royal Institute of Technology
Stockholm, Sweden
Email: ahsanq, janwi@kth.se

Carl During
Micronic Laser Systems AB
Stockholm, Sweden
Email: carl.during@micronic-mydata.com

Abstract—Mechatronic system design is contemplated extensively through model-based approaches. To reflect the multidomain integration inside a mechatronic system during the design process, integration approaches targeting heterogeneous models are essential. An infrastructure supporting the design of multidomain systems is presented. The dependencies between the domain-specific models are managed by utilizing SysML as a common system modeling language between those models. A model integration framework supports model transformations between the system-model and the domain-specific models by utilizing Eclipse Modeling Framework. Structural and parameter type dependencies between different domain-models are handled to provide consistency between different views. This provides an ability to traverse between different views of the system, as well as maintaining consistency between those views. A case study on a robot system is presented to explain the integration of mechanical design model in Solid Edge, and dynamic analysis model in Simulink/SimMechanics through a SysML system model, all under the proposed design infrastructure. The approach is scalable towards other modeling formalisms by building new relations through the SysML model. This will support: co-evolution of the domain-specific models, a better design of multi-domain systems, reduction in modeling in-consistencies, and a reduction in design time.

Key Words: Design Infrastructure, Mechatronics, SysML, Design Methodology, Model Integration

I. INTRODUCTION

A large number of products today require concurrent and multi-disciplinary engineering. At the same time, the constraints on cost and performance of products have become tougher. Trends in a market change at a rapid rate, requiring companies to reduce product time to market. Design time accounts for a majority of time required to realize new products. Therefore, a reduction in design time provides a distinctive competitive advantage to companies for pioneering innovative trends inside consumer markets [1]. The V-model [2] development process lay a foundation for the development of mechatronic systems. However, the design methodologies used within such a process have to be adequately supported by methods and tools for their effective implementation [3]. At the same time, the established design methodologies still need to be better adapted for the design of mechatronic systems.

Design activities for a mechatronic system takes place in different (engineering) domains. Within these domains, the design of the system is represented through models created at different levels of abstraction. A horizontal model abstraction means translation between models that are at same level of abstraction (detail). Whereas a vertical abstraction takes place between models at different levels of abstraction, e.g. between a generic system view and a detailed system view. Since different domain-specific models only provide partial views of the system, there exists dependencies between them. These dependencies arise since one model either contains or affects system properties of another model. These dependencies require going back and forth from one model to the other, performing both horizontal and vertical abstractions, a challenge of model integration. Integrating tools that support synthesis and analysis through these models is another important challenge. This paper is aimed towards development of an infrastructure supporting design of multi-domain systems to achieve a better integration between domain-specific models. As a proof of concept, a set of tools where domain models reside is considered, thus attempting the problem of tool integration.

The remainder of the paper is organized as follows. Section II presents the problem description, and section III discusses related work. Section IV explains the model integration framework which is the backbone of the design infrastructure. A case study is presented in section V, followed by a conclusion and a discussion on future work in section VI.

II. PROBLEM DESCRIPTION

According to Pahl and Beitz [4], and Hubka and Eder [5], engineering design can be divided into four main design phases. The first one is the task clarification phase where the problem is defined and requirements are listed. Second, is the conceptual design phase where essential problems are identified, a function structure is established, and solution principles are searched. The third phase is embodiment design, where preliminary layouts and configurations are selected and evaluated against technical and economic criteria. This is followed by the detailed design phase, where every detailed aspect of the product is specified e.g. dimensions, material, type of sensors, actuators, and controller. From the conceptual design phase, a number of tools support design synthesis and analysis.
Design is different from a mathematical problem whose unique solution is searched [6]. Therefore, many alternative design solutions are considered while going back and forth from detailed-design to conceptual design and embodiment design. This essentially means going back and forth between different modeling formalisms and different abstraction levels supported by different tools, requiring well-integrated models and tools.

For a multi-domain system, tool support becomes specialized starting from the conceptual design phase. For example, mechanical or electrical design concepts can be hand sketched, or can be created inside computer aided design (CAD) tools during concept evaluation. According to Tjalve [7], conceptual design includes putting down the function/means tree, and coming up with a basic product structure, followed by possible quantified structure variants. Here, different types of design analysis are carried out by creation and use of different domain-specific models. The design of the complete system typically becomes distributed among different design groups (utilizing different models/tools), each handling a portion of the design task. However, these groups need to bring their designs and analysis together in order to foresee the design of the complete system. This is sometimes referred to as the design integration complexity.

According to Mrozek [1], the main disadvantage with current tool support is that the tools often work in off-line mode. Therefore, multi-criteria optimization (essential in mechatronic design) is very difficult to implement, as the parameters imported from one tool to the other are fixed. Therefore, integration approaches are vital, providing a capacity to modify system properties in an integrated manner between domain-specific models.

### III. RELATED WORK

In the area of mechanical design and dynamic analysis, a number of research efforts have explained the usefulness of Modelica [8] for multi-disciplinary dynamic analysis of a mechanical system. Engelson et al. [9] explain the use of domain-specific Modelica libraries to perform multi-domain dynamic analysis and controller design tasks on a mechanical system. A partial transformation is presented in [10], where an exchange of data rather than fully autonomous transformation from CATIA [11] (model) to Modelica (model) is proposed, for including CATIA data inside a Modelica model. The same idea is used by Mathworks [12] in a CAD to Sim-Mechanics translator. However, the above-mentioned translators aim to achieve only a horizontal abstraction. Since the design process involves going back and forth between all design phases, a vertical abstraction is also necessary.

When combining different abstraction levels and multiple views into system design, maintaining consistency between those levels and views is important. Adourian et al. [13] addressed the issue of dealing with multi-view consistency. The authors explain how to maintain updates in mechanical design view (in Solid Edge [14]) and its Modelica model through graph transformations.

System engineering approaches focus on designing and architecting the complete system, and also on providing a capability of model re-use. Johnson et al. [15] described utilizing a SysML model [16] for not only adopting a holistic approach towards system design, but also integrating SysML model with dynamic analysis models in Modelica (a vertical abstraction). Paredis et al. [17] proposed the concept of utilizing the same approach towards Matlab/Simulink models. Shah et al. [18] describe the concept of multi-view modeling by creating a SysML profile for an electrical-CAD (ECAD) system, and building model transformations from a domain-specific SysML model to an ECAD model. A similar approach for Modelica is presented in [19] through ModelicaML, a SysML profile for Modelica.

### IV. AN INFRASTRUCTURE FOR INTEGRATED DESIGN AND MODELING

For a mechatronic system, the domains are typically represented as mechanics, electronics, control, and software. Although a modeling language that supports multi-domain modeling can be considered a possible solution, however, it is difficult to cover all design aspects through one modeling language. Therefore, a proposed solution is to utilize different modeling languages, and try to establish relationships between those languages. These relations will support in managing dependencies between models in different domains. This approach aims to provide better tool support during design synthesis (making decision on design alternatives), along with better tool support to perform detailed-design and analysis on the chosen alternative.

#### A. Establishing inter-domain relationships through system-level modeling

While designing a multi-domain system, system-level modeling providing a holistic view is an important part of systems thinking towards design. The system model incorporates findings from the conceptual design stage to create a model of the system. This model can vary from very abstract (during conceptual design), to very detailed as the system design evolves. While designing, the decision on some of the system properties can be made without tool support (e.g., choosing a basic structure of the system). Whereas, other system properties are decided through tool support. Different modeling formalisms (supported by various tools) are utilized in order to solve domain-specific design problems. The resulting domain-specific models provide a partial view of the complete system. We propose that by integrating a complete system view (system-model) with different domain-specific views through an integrated design infrastructure, system design can be better understood. Also, the consistency between different views can be maintained. This essentially bridges the conceptual design phase with the embodiment and detailed-design phase.

There are several modeling formalisms to deal with, hence, it is logical to initially provide proof of the proposed concepts for a small set of formalisms. The proposed approach hence revolves around the following proposition:
Proposition-1: A methodology to integrate one set of modeling formalisms can be extended to integrate other modeling formalisms in the same design infrastructure.

Dependencies arise between the domain-specific models while they are created. However, the domain-specific models do not incorporate these dependencies explicitly, though they have an implicit influence on how these models evolve. It is proposed that each domain-specific model can be related to the system-model by creating a mapping between the corresponding modeling languages. A mapping is a specification of the correspondences between meta-models of two modeling languages. The mapping specifications between SysML and domain-specific modeling languages complements the mapping specifications in between domain-specific modeling languages. Consequently, the system model contains both the system description and the relationships to different domain-specific models. This is shown in figure 1 where the system model serves as a common model between the domain-specific models.

B. Modeling during the product design process

According to Tjalve’s model of the product synthesis process [7], a product design starts with problem analysis. This is followed by creating the main function and sub-functions (with means), leading towards a basic structure. A quantified structure can be obtained by specifying the relative arrangement of elements and dimension parameters, without deciding on element and total form. All of this is done in relation to criteria’s of structure, form, material, dimension and surface. Putting these steps in the context of design phases by Pahl and Bietz [4], these activities fall into task clarification, conceptual design and embodiment design phases. The system model has a strong connection to these design phases, containing information about: requirements, function/means tree, embodiments, and behavior of those embodiments. However, a system-model can only be created after some concepts about system structure and functions have been chosen (quantified structures). Figure 1 shows that system information is modeled in the following section. This is supported by a model integration framework, discussed in the following section.

C. Example models and tools

To further explain the design integration infrastructure, we will now confine ourselves to a small set of heterogeneous models. We consider a mechanical design model in Solid Edge [14], a dynamic analysis model in Matlab/Simulink [12], and a system model in SysML [16].

While utilizing the domain-specific models for performing design synthesis and analysis, two types of changes could appear in the system. Either the system structure changes, e.g. adding a new component. At the same time, a change in system properties could also appear. For maintaining consistency between the domain-specific models, the integration approach needs to support these types of changes in these models. By defining the relationships between SysML and Solid Edge, and between SysML and Matlab/Simulink at the meta-model level, it is possible to maintain consistency between the three models. This is explained further through a robot example in section V. This approach extends the approach presented in [9] by utilizing a system model for providing a vertical abstraction, whereas CAD to Modelica translation only provides a horizontal abstraction. The approach also extends [15], [17], [18] by inclusion of mechanical CAD models along with dynamic analysis models.

To perform transformations between Solid Edge, Simulink, and SysML models, a platform is required where the meta-models, mapping, and transformation rules can be defined. This is supported by a model integration framework, discussed in the following section.

D. Model integration framework

The views provided by the domain models and the system model exist at different levels of abstraction. Therefore, a mechanism that supports abstraction between views (models) in order to maintain consistency is required. Model transformation is an automated process of converting a source model into a target model, through defined transformation rules [20]. The transformations are defined at the meta-level, based on meta-models. A mapping explains the link between the source meta-model and the target meta-model. The mapping is used to create transformation rules, which are executed by a model transformation engine.

The objective of the model integration framework is to represent modeling formalisms of different domains in a common formalism, hence facilitating information exchange between them. This requires a platform where integration activities between various tools can be performed. Eclipse Modeling Framework (EMF) [21] is one such platform, where
a set of tools can share data and semantics. Inside EMF, different models are defined as EMF Ecore models. A meta-model each for Solid Edge, Matlab/Simulink and SysML are defined as an Ecore model. These meta-models are called integration models, and they support exchanging information that represent dependencies between the models. Figure 2 shows the Solid Edge meta-model, with Assembly and Part as main elements. A Part which is an occurrence inside an assembly, contains variables (e.g. material, and geometric properties). There can be different relationships between parts, shown as Relations3D.

In order to have constructs that closely represent a mechanical system inside Simulink, the SimScape/SimMechanics toolbox is considered a relevant choice, leading towards the creation of a SimMechanics meta-model as shown in figure 3. The SimMechanics meta-model defines a SimMechanics model as a group of systems, each represented by a MechanicsElement, and connected to each other through ports. Currently, only main SimMechanics library constructs are considered in the meta-model. Further work attempts to integrate this meta-model with the Simulink meta-model as developed in [22].

SysML is a general purpose systems modeling language, enabling modelers to understand the complete system design. Consequently, this means that SysML lacks formal and detailed semantics needed to represent the domain-specific information. However, SysML provides a procedure for creating domain-specific semantics through profiles. A profile extends the SysML meta-model through explicit constructs that relate closely to domain specific language constructs. Since this extension is done at the meta-level, automated model transformations are facilitated by building mappings based on language constructs rather than variable names [18]. Profiles also support assigning pre-defined properties to blocks, such as material type for a Solid Edge Part.

A different approach from profiling is to construct a SysML model with its own constructs, and embedding the domain-specific information inside the model transformation. For example traceability information through a model transformation [23]. However, this might be cumbersome if more than one construct from a domain-specific language relates to the same SysML construct. For example, an Assembly and a Part inside a mechanical CAD model can both relate to a SysML block. In this case, relations such as an association between Assembly and a Part will provide the necessary differentiation between the two blocks. On the other hand, the SimMechanics meta-model differentiates between e.g. a body, and a joint. Since a body and a joint will both be represented as a SysML block, it will not be possible to differentiate between them while transforming between SysML and SimMechanics. Therefore, we propose to utilize SysML profiles for mechanical-CAD (MCAD) and SimMechanics. The MCAD profile contains stereotypes such as Assembly, Part, Body, Joint etc., to be able to perform an appropriate model transformation between SysML and Solid Edge. Similarly stereotypes are applied using SysML profile for dynamic analysis (DA) using Simulink/SimMechanics. Both the profiles are omitted here due to space concerns.

Mapping relations are defined between SysML and Solid Edge, and between SysML and SimMechanics meta-models. For example, a Solid Edge Assembly (in Solid Edge meta model) maps to an Assembly in the SysML profile for Solid Edge, and vice versa. Similarly, mapping relations are defined between SysML and SimMechanics.

Parsers are required to extract/export the relevant information from/to the tool, and populate the corresponding Ecore model. A parser is built to extract CAD data from the Solid Edge model. To further classify the information as representing dependencies in a Solid Edge model, a parametrized model is utilized. This parametric model is based on the notion of driving variables, controlling various properties inside a CAD model. The parser is capable of reading a model assembly inside Solid Edge, and extracting the driving variables for each part in the assembly. The parser writes that information as an XML document, complying with the Solid Edge meta-model. The same procedure is adapted while traveling towards the
Solid Edge model. Figure 4 shows how the Solid Edge parser is used.

In order to read a Simulink/SimMechanics model into an Ecore model, Simulink connector provided by Cameo Workbench [24] (Eclipse environment) is utilized. The connector populates a Simulink Ecore model, complying to the Simulink meta-model inside CameoWorkbench. The same connector allows creating a Simulink/SimMechanics model through an Ecore source. CameoWorkbench provides another connector to MagicDraw [25], which populates an Ecore model complying to a UML2 meta-model (with SysML profile). The connector also provides an opportunity to create a MagicDraw model from an Ecore source that complies to UML2 meta model. This connector is utilized to read/write SysML model with applied stereotypes for MCAD, and DA. Figure 4 provides a complete overview of the corresponding connector for Simulink/SimMechanics, Solid Edge, and MagicDraw (SysML).

After populating Ecore models from Solid Edge, SimMechanics, and SysML, a final step is building bi-directional transformations between them based on defined mappings. These transformation rules are written in Atlas Transformation Language (ATL) [26] and Model Query Language (MQR), which is an imperative model transformation language supported by CameoWorkbench. Figure 4 shows the transformations between the Ecore models.

V. CASE STUDY

To exemplify the proposed approach, a two degree of freedom robot system is considered a relevant example. As mentioned in section IV, it is assumed that a concept about system structure has been chosen based on identified function/means. This provides information about requirements for the robot system, a decision on a basic structure, followed by information about variants of the quantified structure. One such variant is presented here to perform further analysis on element forms, and the total form of the robot system.

Figure 5 shows a quantified structure of the robot, stating that the robot is composed of two links, two revolute joints driven by a DC motor each (see figure 5). The rotation of each motor is measured by an encoder. Figure 5 also shows that a decision on relative arrangement of main components has been made according to requirements on the robot system. At this point, the modeling/analysis process can start to further design the embodiments. For this purpose, a mechanical design model of the robot is created in Solid Edge, a dynamic analysis model is created in Simulink/SimMechanics, and a system model is created in SysML. Figure 6 shows the different types of information (supported by different views in SysML) contained in the SysML robot model. The Solid Edge and Simulink/SimMechanics models are represented by the information that they require, and information that they update after design iterations.

Let us consider two examples to explain how consistency
between the three views of the robot is maintained. The link length of arm A (L_A) could be modified during dynamic analysis in Simulink/SimMechanics. The integrated infrastructure ensures that (L_A) inside SysML and Solid Edge is consistent with this change. Another scenario is where design synthesis suggest to add a new link (L_C) to the robot (a structural change). Adding L_C in the SysML model results in a new part appearing in the Solid Edge model, since (L_C) is recognized as a new part through the SysML MCAD profile. On the other hand, changing the dimension of current link (L_A) is a change in property only.

VI. CONCLUSION

This paper presents an infrastructure to support integrated design and modeling of multi-domain systems. The infrastructure utilizes SysML as a common language between other modeling languages, to support strong relations between system design and domain-specific design. Mappings between SysML, Solid Edge and SimMechanics meta-models, specification of declarative transformation rules between meta-models through ATL, and the model integration framework utilizing EMF are important constituents of this mechatronic design infrastructure. Although only MCAD models and dynamic analysis models are considered in this paper, through meta-modeling, the approach may be extended towards other modeling formalisms. The robot case study highlighted that dependencies between domain-models have a strong connection to the design-context, in this case it was position control of the robot. Other aspects, e.g. temperature effects can play a role as dependencies if they influence the fulfillment of the system requirements. By utilizing parametrized models in Solid Edge, different design aspects can be accounted for.

SysML provides good ability to define abstract relationships between system requirements, structure and behavior. Through the proposed infrastructure, consistency between the SysML model and other domain-models can be maintained, as these models co-evolve while the design proceeds through different design phases. Consistency management is an important goal for the design infrastructure to be useful for designers during different product design phases. However, as design research suggests, conceptual design is a very dynamic phase in terms of number of, and change in concepts. The extent to which this infrastructure will be useful during product conceptualization has yet to be evaluated through future work, where the infrastructure will be utilized for a few example systems. An empirical study to gain designers opinion while utilizing this infrastructure is also important for evaluating the real benefits. We aim to contribute towards a design methodology for mechatronic systems, by extending current design procedural models towards multi-domain design.

ACKNOWLEDGMENT

The authors would like to thank Carl-Johan Sjostedt and Alex Schenkm for their valuable comments and suggestions.

REFERENCES

Appendix D

Designing Mechatronic Systems: A Model-Integration Approach

Ahsan Qamar, Jan Wikander and Carl During, Designing Mechatronic Systems: A Model-Integration Approach, Accepted for Publication in 18th International Conference on Engineering Design (ICED11), Copenhagen, Denmark, 2011
DESIGNING MECHATRONIC SYSTEMS: A MODEL-INTEGRATION APPROACH

Ahsan Qamar¹, Jan Wikander¹ and Carl During²

(¹) KTH-Royal Institute of Technology, Sweden (²) Micronic Laser Systems, Sweden

ABSTRACT

Development of mechatronic products requires different types of design models in order to support both domain-independent specifications and domain-specific principles. This research aims to find out how system-level modeling can support mechatronic design, and how the integration of system-level modeling and domain-specific modeling can be supported during different design phases. A design example of a hospital bed’s propulsion system is presented to show firstly the relationship between conceptual design and system-level modeling, and secondly the need for integration of system level and domain specific design models. An integrated modeling and design infrastructure is proposed to support abstraction between mechatronic design models, hence supporting co-evolution of design models. The paper concludes that a mechatronic design problem can be better supported through such an integrated design approach. However, usability of this approach needs to be further supported by more case studies in the future.

Keywords: Mechatronics, system design, design infrastructure, model integration

1 INTRODUCTION

Technical systems today have to support a large number of functions at reduced cost. The efficiency and cost effectiveness gained by implementing the required functionality through electronics and computer software has become a major driver towards development of mechatronic products. These products are characterized by increased integration of mechanics, electronics, and computer software, requiring companies to establish cross-disciplinary teams consisting of several domain-experts. Tomiyama et al. [1] state that multi-disciplinary product development (as in mechatronics) introduces difficulties such as the need for an inter-disciplinary design language, how to deal with different stakeholders, and how to deal with inter-disciplinary design problems. Mechatronic design requires a collaborative effort between different domain experts within the cross-disciplinary team. However, each domain contains different collections of design methods and tools. An overall mechatronic design method is yet to be conceived. As engineers do not necessarily possess cross-domain knowledge, it is difficult for an individual to understand the inter-disciplinary problems, especially in the context of complex products. Therefore, communication between involved domains is essential throughout different design phases in order to avoid integration problems in mechatronic product development. In addition to this, technological advancements and increased functionality contribute to high complexity in designing mechatronic products.

Modeling is an important design activity, where modelers try to gain information about consequences of their decisions early in design. Buur and Andreasen [2] state that the success of a mechatronic design project depends especially on the ability of the designers to communicate and visualize their ideas to the rest of the group. Design models permit a designer to describe his or her thoughts for better understanding, both individually and by the group. Depending on the design stage, a design model can be abstract or detailed. A design model should be carefully developed to model only the product properties necessary at the current design stage [2]. This restriction in scope is necessary: firstly, since information about a design problem increases through different phases of design, secondly, because a design model with too many product properties will become unnecessarily complex to serve the purpose for the designer. Hence, product design is based on different design models reproducing different product properties. In a mechatronic design scenario, design models vary between different domains. Some of these models define and describe the product from the domain perspective such as mechanics or electronics; others are used to evaluate product properties within a domain such as dynamic analysis of a mechanical design.
The partitioning between different mechatronic domains is laid very early in product design. By defining function principles to the function structure, the designer allocates different technologies to the product function/sub-functions (design concept). However, this often leads to an isolated development within a domain, and optimization of the individual modules, rather than the complete system. Buur et al. [2] and Gausemeier et al. [3] state that specifying a mechatronic design concept requires a new design model. Buur et al. [2] define the new model types to support abstract function structure independent of a technology, function principles supported through different technologies, and specification of the interfaces between different technologies. Gausemeier et al. [3] utilize a new modeling language for supporting abstract function structure, domain-spanning function principles, and interfaces.

One of the main ideas behind the research treated in this paper is to utilize design models that are suitable to capture domain-independent specifications. At the same time, the designer can apply multi-technological function principles to the design concept. The domain-specific models evolving from these multi-technological function principles can be integrated with the domain-independent model, the other main proposal of this paper.

Since design evolves not only from one design phase to the next, but also in between domains, it is important to support abstraction between models of different engineering domains. Moreover, it is also important to keep the design models consistent with each other if a certain design model is modified at a certain design stage. One approach towards achieving this abstraction is through model transformations. Some examples of integrating design models through model transformations are presented in [4] and [5]. Consistency between mechatronic design models has been discussed in [6]. While these approaches extend the software engineering principles towards model-based development, this paper aims to explain the relationship between design models within a mechatronic design problem, and proposes a solution to multi-domain model integration. The paper utilizes a design study of a servo-propelled hospital bed performed at the Department of Machine Design at the Royal Institute of Technology, Sweden, to answer questions as such:

1. How to establish an initial domain-independent design specification through a design model, and obtain a complete system view required in a mechatronic design problem?
2. How to establish relationships between domain-independent and domain-specific design models?
3. How to integrate different design models developed at different design stages?

The remaining part of the paper is structured to answer the above questions as follows. Section 2 discusses model-based design in relation to engineering design methods. The section proposes a solution for solving the communication problem between domain experts in a mechatronic context (question 1). Section 3 presents the design study of the hospital bed system highlighting the conceptual design phase where domain-specific design models were also utilized. Section 4 is about answering the question: how to integrate models (question 2), and how better design can be achieved through model integration, supported by a small example (question 3). Section 5 concludes the paper, including a discussion on proposed future work.

2 MECHATRONIC DESIGN AND MODEL-BASED DEVELOPMENT

Model-based engineering (MBE) is about elevating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system [7]. The systematic design approach from Pahl & Beitz [8] shows three main product design phases: conceptual design, embodiment design, and detailed design. During these design phases, models increase in detail with the passage of time, and abstraction between models needs to be supported in order to manage the modeling process as a whole.

Design activities for a mechatronic system are typically performed by a multi-disciplinary team of domain experts. Different design methods (available in different domains) are followed by the domain experts, who are unlikely to possess inter-disciplinary knowledge to get a detailed enough understanding of the whole design problem. Therefore, it is difficult to establish a common mechatronic view; rather different domain views are established and the dependencies in-between are not clear. Hence, it is necessary to establish some means of communication between such views in order to avoid integration problems. Frey et al. [9] classify the communication between two design-domains in terms of communication possibilities between persons, between methods, between models, and between analysis tools. We argue that for a mechatronic design problem, two possibilities can be undertaken to attack the communication problem:

---

The remaining part of the paper is structured to answer the above questions as follows. Section 2 discusses model-based design in relation to engineering design methods. The section proposes a solution for solving the communication problem between domain experts in a mechatronic context (question 1). Section 3 presents the design study of the hospital bed system highlighting the conceptual design phase where domain-specific design models were also utilized. Section 4 is about answering the question: how to integrate models (question 2), and how better design can be achieved through model integration, supported by a small example (question 3). Section 5 concludes the paper, including a discussion on proposed future work.

2 MECHATRONIC DESIGN AND MODEL-BASED DEVELOPMENT

Model-based engineering (MBE) is about elevating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system [7]. The systematic design approach from Pahl & Beitz [8] shows three main product design phases: conceptual design, embodiment design, and detailed design. During these design phases, models increase in detail with the passage of time, and abstraction between models needs to be supported in order to manage the modeling process as a whole.

Design activities for a mechatronic system are typically performed by a multi-disciplinary team of domain experts. Different design methods (available in different domains) are followed by the domain experts, who are unlikely to possess inter-disciplinary knowledge to get a detailed enough understanding of the whole design problem. Therefore, it is difficult to establish a common mechatronic view; rather different domain views are established and the dependencies in-between are not clear. Hence, it is necessary to establish some means of communication between such views in order to avoid integration problems. Frey et al. [9] classify the communication between two design-domains in terms of communication possibilities between persons, between methods, between models, and between analysis tools. We argue that for a mechatronic design problem, two possibilities can be undertaken to attack the communication problem:
1. A mechatronic design methodology could be followed. VDI2206 [10] introduces such a design methodology. However, it does not cover the management of dependencies between mechatronic design domains. Also, the means to support abstraction between system-design and domain-specific design (per VDI2206) are lacking. A vertical abstraction adds detail to a model or reduces it, while a horizontal abstraction typically takes place between models of the same detail, often in different domains. As highlighted by Buur et al. [2], new mechatronic design models are needed that support both domain-independent and multi-domain modeling capability. Some modeling languages, such as Modelica [11] and MapleSim [12] support the creation of multi-domain design models (spanning different stages of VDI2206). However, domain-independence is also important for the different domain-experts to share a common product view. We conclude here that a single multi-domain modeling language cannot provide a solution for mechatronic design problems, a conclusion also supported by Shah et al. [5]. Such a language is difficult to develop, support, and evolve with time.

2. A framework that supports mechatronic design can be utilized. Here, we can let the domain experts utilize the existing methodologies inside each domain, and provide means of communication between domains through the framework. As suggested by Frey at al. [9], a person-to-person and method-to-method communication is either error-prone, or not directly possible. Even though the communication between analysis tools is possible, however, it is based only on execution of design models and not their development. Therefore, a model-to-model communication is employed during this paper, which allows for both vertical and horizontal abstraction during model-based development.

For communication through multi-domain models in a perspective of engineering design, an infrastructure is required, supporting evolution of design models while design proceeds between different design phases. The infrastructure should support development of domain-independent design models and specification of interfaces between design domains, development of domain-specific design models, and integration between all these design models. Such an infrastructure is vital to integrate multi-domain models, which is a key proposal of this paper, as explained in Section 4. Conceptual design is a dynamic phase in terms of change in design and interaction of designers. Initial product synthesis – being part of conceptual design – serves as a basis for developing an abstract function structure and corresponding function principles. In a model-based development with model as the primary artifact, it is important to capture the product synthesis information inside models. In agreement with Gausemeier et al. [3], we will use the term common specification language to denote a common platform for different domain experts to define and specify a system. Figure 1 shows that a system-model can be developed through a common language, based on the findings of product synthesis phase. Here, the interfaces between chosen design concepts can be specified starting in a black-box manner and continuing towards increasing detail. The Systems Modeling Language (SysML) [13] is a general purpose modeling language where a modeler can specify a system to a level
that enables its association to other design models [14]. We utilize SysML to represent the multi-domain function principles through generic SysML constructs. The resulting model can be utilized as a common design model, by building relationships with other design models. The relationships can be built by utilizing the extension capability in SysML to create domain-specific parts of the system model (Figure 1), where the parts are built with concepts of a particular domain. The proposed integration infrastructure allows integration of the domain-specific design models with the domain-specific parts of the system model. Hence, as the design progresses through domain-specific design models, the system model also increases in detail. Design iterations continue to take place between different design stages, and design models consistently evolve through the integration infrastructure (Figure 1). Another approach is to first create domain—specific design models based on product synthesis (red arrows in Figure 1). The resulting design models can be related with each other by transforming them into the system model through the integration infrastructure. We do not strictly propose to follow this order or the order starting with the system model, to leave flexibility for future work.

3 DESIGN EXERCISE: A SERVO-PROPELLED HOSPITAL BED

This section is based on a mechatronic design example in order to better reflect on the following questions (rephrased from questions: 1, 2, and 3 in section 1):

- How to establish relationships between domain-independent and domain-specific design models?
- How to integrate different design models developed at different design stages?

By performing a design exercise on the hospital bed example, we try to identify the needs for the integrated modeling and design infrastructure, and determine how dependencies can be managed through the SysML model. The aim with the exercise is to design an active (driven) wheel module which can be utilized on common hospital beds. The design activities performed within the conceptual design phase are explained in the subsequent sections.

3.1. Conceptual design

During conceptual design, a group of designers were provided with a set of requirements for the hospital bed. The design team consisted of six team members with backgrounds in mechanical engineering, electrical engineering, control engineering, system engineering and computer science. The requirements for the design of the wheel module were classified in different categories on a white board, and used as a reference throughout the initial conceptual design phase. Some of the main requirements for hospital bed are presented below; the rest is omitted for space concerns:

- Bed speed of 2 m/sec on a maximum 5° slope (power, driving requirements)
- Being able to turn on the spot and move in any direction (driving configuration, control)
- Wheel solution packaged as one module (sensor, motors, control, wheel module configuration)

Based on these requirements, the design group discussed the following main points:

- Which configuration of driving and steering (number and placement of active wheels: propulsion system configuration) will provide the best solution in terms of drivability, steering, and cost effectiveness. Six different configurations were discussed.
- How to provide a modular solution, yet one that works on current hospital beds. This includes decision on sensors (wired/wireless), power required to move maximum load, battery (central or distributed), battery charging scenarios, and mechanical interfacing.
- Configuration of driving and steering within one wheel module assembly (wheel module configuration). Two configurations were discussed, knowing available wheels and information about drive and steering actuators. The braking requirement for each wheel was also discussed to be accounted in the solution.
- A central controller is required to control the driving and steering of the bed as a whole, by controlling the available drive and steer actuators in each wheel module assembly.
- Safety considerations in relation to the use of wireless devices, patient safety and comfort requirements.

During the initial conceptual design phase (lasting for 5 hours), an initial decision was made on some of the concepts. However, some concepts needed further analyses before any decision about them could be made. Some of these concepts were:

- Wheel-module configurations
- Analysis of driving, steering and braking actions in relation to bed movement for six different propulsion system configurations, and two different wheel module configurations
- Controller complexity for different wheel module- and propulsion system- configurations
- Which battery is suitable for the bed, i.e. an analysis of power to weight ratio and cost considerations of batteries
- Relationship of distributed battery and central battery in connection to wired and wireless sensors, and the cost difference for each concept

Different propulsion system configurations led to six possible design alternatives, all providing driving and steering capability. These concepts were compared in a weighted decision matrix (omitted here due to space concerns) against a set of basic design criteria. The decision matrix showed that the propulsion system configuration based on either a diagonal configuration or a fully configurable configuration as the most feasible options. The same result is modeled inside a system-model (Figure 3(b)) which is discussed in the following section.

3.2. System design/modeling
After completing the initial conceptual design, the requirements for the wheel module along with the possible solution concepts were known (working structure). A system model was built based on these findings. The suitability of SysML as a modeling language during the conceptual design phase has been documented in [15] and [16]. This section will present a few snapshots of the SysML model, especially its relation to the performed synthesis, and to the creation of domain-specific models.

Figure 2. (a) Use cases showing manual and propelled operation of bed. (b) Top-level overview of bed showing main actors. (c) A set of requirements for the power subsystem

Figure 2(a) shows the manual and automated driving use cases performed by the bed operator. The accelerate, brake, and steer use cases are performed by the operator manually. The top-level structure of a hospital bed is shown in Figure 2(b), containing the main actors and the environment in which the bed operates. The system HospitalBed contains parts (regardless of technology) mentioned in the parts compartment. Figure 2(c) shows some of the derived requirements, in this case for the power subsystem. All models in Figure 2 are developed based on the information obtained through the product synthesis phase of the design exercise. The main structure of the hospital bed propulsion system is shown in Figure 3(a), showing different subsystems. Each subsystem contains a combination of multi-domain components, represented through general purpose semantics. The interfaces between different system components (represented by generic constructs) can also be specified. The ChassisSubsystem in Figure 3(a) contains two to four WheelModule blocks. This is based on the conceptual design phase discussion deciding that there will be a minimum of two active wheels on each bed, and each wheel will be enclosed as one complete module with driving and steering capability (Modular Components Constraint). The six alternative propulsion system configurations are
compared against stated criteria by a weighted objective function \((\text{WheelModuleObjectiveFunction})\) used inside the \text{WheelModulePerformance TradeOff} block (Figure 3(b)).

During the conceptual design phase, a discussion was made about off the shelf wheel-motors, only requiring a steering mechanism to provide a steering and driving capability in one module. Based on the wheel-motor, two configurations for the wheel module assembly were discussed, as represented in Figure 4(a). Each configuration consists of a drive motor-wheel unit and a steering actuator/transmission as constituent parts. Interfaces \text{Drive} and \text{Steer} are \text{provided interfaces} to each configuration, letting the drive motor and steer configuration to interact with the wheel-module configuration. Interface \text{Sensor} (to measure angle and velocity of wheel) is a \text{required interface}, letting the wheel configuration send sensor measurements to other blocks. Configuration 1 consists of a gearing with a steer motor to steer the drive motor assembly (Figure 4(b)). Configuration 2 contains freely revolving driving assembly with the drive wheel mounted off the vertical rotational axis of the steering assembly. A brake modulator inside the steer assembly locks the drive assembly at the required angle. Position encoders were selected to measure the steering and driving rotation angles.

In order to make a decision about the wheel module configuration, it was important to realize each configuration in terms of form, and analyze them in terms of behavior. For this purpose, a CAD model and a dynamic analysis model for each configuration were built, as discussed in section 3.3. For the controller, a centralized architecture and a distributed architecture was considered. Although both the architectures were also dependent upon selection of wireless or wired transmission, it was decided to control the angle and velocity of each wheel through a local controller (Figure 5(b)). If a wireless transmission is considered, a local battery can power the wheel module and the local controller. In this case, a wireless transceiver would send data to the main controller and receive reference commands from it as shown in Figure 5(a). Figure 6 shows the complete distributed control architecture with two wheel modules and the user interface inputs. It can be noted that in case of wired...
transmission, the interface WirelessData and WirelessCmd will be replaced by wire connections. In this case, a central battery can also be considered relevant instead of a localized battery.

The system-level modeling through SysML proved to capture the information gathered during the synthesis phase effectively, as displayed through different SysML diagrams. Moreover, all SysML diagrams presented here are consistent with each other. This means that introducing a change in one diagram during design leads to relevant changes in other diagrams too. The SysML tool (MagicDraw [20]) supports keeping different SysML diagrams consistent with each other.

3.3. Domain-specific design

As discussed earlier, the decision about some of the components, configurations, and alternatives needed further analyses through domain specific models. This section provides a limited overview about mechanical design and analysis of the wheel module, in order to make a decision about the wheel module configuration assembly and the relevant components. Other issues such as controller bandwidth, energy supply and communication between wheel modules etc. were also discussed. However they are not presented here due to space concerns.

**Wheel module assembly design**

The CAD modeling of the two wheel module configurations was performed to get an estimation of the size of the whole module and to know the necessary components. The wheel-motor model was utilized to construct the assembly. The two concepts can then be compared based on manufacturing cost, safety, and performance criteria. Figure 7 (a) and (b) show the two configurations.

**Analysis of wheel module configurations**

In order to analyze how the bed moves with each wheel module configuration, a dynamic analysis model was created. The multi-domain physical modeling and simulation tool MapleSim [12] was used
to construct an initial simplified model of the two concepts. This analysis also highlights the control complexity of each configuration. Figure 8(a) shows the rigid body model using configuration 1. Figure 8(b) shows the rigid body model using configuration 2 with the brake modulator represented as a clutch that locks or releases the drive wheel. Both configurations gave acceptable performance. However, configuration 2 (i.e. utilizing a brake modulator to provide steering) requires an intelligent control strategy to control steering angle, whereas configuration 1 is a much simpler control problem.

Figure 7. CAD models of wheel module configurations. Both configurations contain the same drive motor and wheel (a) Configuration 1 with steering motor for steering (b) Configuration 2 with brake modulator for steering

Figure 8. Design models created to analyze the bed movement. (a) Configuration 1 (b) Configuration 2, the clutch mimics the brake modulator. A planar force in each driven wheel mimics the steering and driving actuator in both configurations

The above design and analysis through domain-specific design models was necessary in order to take a decision about wheel-module configuration – a situation which is typical to most mechatronic system developments. This further emphasizes the need to integrate the system-model with these domain specific design models in order to ensure consistency while designing. Section 4 throws further light on this topic.

3.4. Dependencies between domain-models

Through the SysML model, the design concept for the propulsion system was specified. Further domain-specific design provided more information in order to make decision about sub-systems, components, and alternatives. The domain-specific models were created based on information obtained from the SysML model. For example, the CAD model in Figure 7(a) was made based on the initial sketches made during conceptual design phase, which led to creation of figure 4(b). The information about wheel, wheel-motor, steering motor and steering configuration was needed before the CAD model could be created; a dependency between the SysML model and the CAD model. Moreover, the dynamic analysis model in Figure 8(a) was created based on the propulsion system configuration, a dependency between CAD model, SysML model and the dynamic analysis model. Since the SysML model contains the wheel module requirements and the complete system structure, it is important to keep this model consistent with other design models. A change in e.g. the CAD model
during design iteration should be traced back to the corresponding model element inside the SysML model, a problem of multi-view consistency as explained in [5]. The following section presents our proposed infrastructure for integrating models.

4 INTEGRATING MODELS FOR ABSTRACTIONS IN DESIGN

Several iterations may take place both between the design phases and between the corresponding design models before the design process is complete. Each model contains a certain level of detail. For example, SysML provides a rather abstract system view, whereas, a CAD model provides a detailed view. Figure 9(a) shows a domain-independent system-model, and the corresponding domain-specific models that are typically created during design.

![Figure 9. (a) Iterations between multi-domain design models, and abstraction during different design phases (b) Integrated modeling infrastructure based on EMF](image)

Keeping the domain models consistent with each other requires means to manage inter-domain dependencies. A domain-model contains information that has relation to other domain-models (dependencies). It also contains information that is relevant only within the domain-model itself. We utilize a meta-modeling approach to characterize the information relations between domain-models. A meta-model specifies the abstract syntax of a modeling notation [17]. A meta-model for a domain specifies the concepts that exist within that domain [5]. Therefore, different domain-models comply with different meta-models. By defining relationships between these meta-models, it is possible to write transformations between models, which comply with those meta-models. We utilize the Eclipse Modeling Framework (EMF) [18] to define the meta-model of each domain, and we define relations between meta-models through a rule-based language (ATL) [19]. Figure 9(b) shows the integrated modeling infrastructure, in this case involving two domains: mechanical CAD, and dynamic analysis. Inside EMF, a model is represented as an Ecore model. Therefore, each domain is specified through a domain-meta model in Ecore, e.g. MCAD Ecore meta-model (Figure 9(b)).

Since SysML is a general purpose modeling language, the aim of using SysML is to create a system model independent of any domain or technology. However, our proposition to establish relationships between domain-models through SysML requires SysML to support domain-specific concepts. This has been supported in SysML through creation of profiles. A profile extends the SysML meta-model with the needed domain-specific constructs. Figure 9(b) shows two profile blocks, each relating to a domain-specific model. For example, the SysML MCAD profile model contains a model built with MCAD concepts such as: assembly, part, relations etc. An MCAD-specific SysML meta-model can then be created as an Ecore model. For dynamic analysis (DA) models, a DA-specific SysML meta-model can be created. Relationships between the SysML model and the domain-specific part of the SysML model are necessary to establish a link between the system design and the domain-specific design. This can be achieved by allocating (manually) elements/components in SysML model to their counter parts in domain-specific part of the SysML model. In this way, the relationships between different design models can be established (inter-domain dependencies) through the SysML model, by
establishing allocation relationships between SysML model and the corresponding domain-specific parts of it. For example, in Figure 9(b), it is possible to establish relationships between the MCAD model and the dynamic analysis model (DA) by establishing relationships between the SysML model and the SysML MCAD Profile Model, and between SysML model and the SysML DA Profile Model. These relations can be manually drawn, or can be automated. An automation procedure requires relating a concept in one meta-model to a corresponding concept in the other meta-model and writing transformation rules based on those relations. Figure 9(b) shows an ATL transformation between each domain model and the domain-specific part of SysML model in EMF.

4.1. Integration Example

This section will illustrate an integration example between the MCAD model of the hospital bed wheel unit and the SysML system model. Figure 10(a) shows a generalized meta-model for MCAD, based on constructs such as assembly, part, variable, relation etc. There are differences between MCAD tools, which can lead to a different meta-model for each tool. However, we propose a generalized meta-model to be adapted for a domain such as MCAD, and aligning different modeling languages within the domain to that meta-model. An important consideration here is that the proposed MCAD meta-model does not contain all MCAD concepts; rather it only contains constructs relating to the type of information that we are interested to obtain from MCAD tools.

A model complying with the MCAD meta-model can be created by making API calls to the MCAD tool of choice. We utilize Solid Edge [21] in this example, and populate a Solid Edge Ecore model (representing the MCAD Ecore model) through a developed parser (see Figure 9(b)). The same parser allows us to create a model inside Solid Edge based on an Ecore model. Hence, it is now possible to represent the wheel module assembly configuration 1 as an Ecore model shown in Figure 10(b). The SysML profile for MCAD is based on MCAD concepts, which are extended from the SysML metamodel elements. For example, a Part extends a SysML block etc. Figure 11(a) shows SysML4CAD profile meta-model represented in Ecore. Another parser is used that populates an Ecore model (see Figure 9(b)). Figure 11(b) shows the wheel module configuration 1 as SysML4MCAD Ecore model. Declarative ATL rules can be specified for each meta-model construct, to create a target model element based on the source model element. This completes a transformation between MCAD and SysML4CAD. In a similar fashion, transformations will be written between the DA model and SysML4DA model based on their corresponding meta-models as shown in Figure 9(b). At the end, the
relationships between SysML4CAD profile model and SysML4DA profile model can be specified, to describe the dependencies between the two domain specific design models.

![Diagram](Image)

**Figure 11.** (a) SysML profile for CAD (SysML4CAD) in Ecore (b) Wheel module configuration 1 represented as an instance of SysML4CAD profile

### 4.2. Mechatronic procedure, domain models under SysML Umbrella

The extended system model allows us to first establish an overall overview of the mechatronic system, to specify the function principles, and to specify the principle solution through a common specification language. Through the model integration framework, domain-specific views are integrated, and dependencies and consistency between design models are maintained via the system model. We believe that establishing abstract system information through the system model and keeping other design models consistent with the system view provide good support for identifying inconsistencies among design models, and avoiding integration failures as a result. The integration infrastructure can in this way support development of better mechatronic design solutions.

### 5 CONCLUSION

In this paper, a proposal for mechatronic design infrastructure based on the integration of design models is presented. Mechatronic product development requires a common specification language for different domain experts to communicate with each other. It also requires design models that can support multi-domain constructs inside one modeling language to be able to model a mechatronic concept. Though some modeling languages support multi-domain modeling and analysis (such as Modelica), other design models are very domain-specific. Using SysML to establish a domain-independent system model, and establishing relationships and means for automated integration with other design models is the main theme of this paper. The paper presents a step by step construction of a SysML model and some domain-specific models for the design problem of a hospital bed’s propulsion system. System level modeling can play a major role in mechatronic product development, thus it has to be supported through all development phases. The proposed integration infrastructure enables us to maintain both the system model and domain-specific design models throughout the product development process. A small integration example between an MCAD model and a SysML model is presented to exemplify our proposal. Future work targets extending the model-integration example towards a more comprehensive integration example between: SysML, MCAD, and dynamic analysis (DA), evaluating the support potential of this approach during mechatronic design phases.

**ACKNOWLEDGEMENT**

The authors are thankful to Carl-Johan Sjöstedt, Daniel Frede, Daniel Malmquist, Hamid Shahid, and Mohammad Khodabakhshian for their valuable inputs in performing the hospital-bed design-study.
REFERENCES


Contact: Ahsan Qamar
PhD, Candidate
KTH- Royal Institute of Technology,
Division of Mechatronics, Department of Machine Design,
Brinellvägen 83, 10044, Stockholm, Sweden
Email: ahsanq@kth.se, Web: www.md.kth.se/~ahsanq
Appendix E

Mechatronic Design- Still A Considerable Challenge


Copyright © 2011, ASME.
ABSTRACT

Development of mechatronic products is traditionally carried out by several design experts from different design domains. Performing development of mechatronic products is thus greatly challenging. In order to tackle this, the critical challenges in mechatronics have to be well understood and well supported through applicable methods and tools. This paper aims at identifying the major challenges, by conducting a survey of the most relevant research work in mechatronic design. Solutions proposed in literature are assessed and illustrated through a case study in order to investigate, if the challenges can be handled appropriately by the methods, tools, and mindsets suggested by the mechatronic community. Using a real world mechatronics case, the paper identifies the areas where further research is required, by showing a clear connection between the actual problems faced during the design task, and the nature of the solutions currently available. From the results obtained from this research, one can conclude that although various attempts have been developed to support conceptual design of mechatronics, these attempts are still not sufficient to help in assessing the consequences of selecting between alternative conceptual solutions across multiple domains. We believe that a common language is essential in developing mechatronics, and should be evaluated based on: its capability to represent the desired views effectively, its potential to be understood by engineers from the various domains, and its effect on the efficiency of the development process.

1 INTRODUCTION

The design of mechatronic products is a multidisciplinary activity and is performed to attain product related advantages, which cannot be obtained by mono-disciplinary efforts. Along with the benefits from having several engineering disciplines involved in the design activity, complexity of the task increases accordingly. Since a mechatronic product is composed by solutions from the areas of mechanics, electronics, and computer software, special attention has to be paid to dependencies in the product and between the design activities. A lack of sufficient attention to the dependencies causes integration problems and increased development cost [1].

The aim of this paper is to gain a good understanding of the challenges related to design of mechatronics (referred to as mechatronic challenges hereafter), in order to help improving the development of solutions for mechatronic designers. A systematic and a thorough literature review is carried out to determine the mechatronic challenges and their proposed solutions as presented by researchers. The remaining part of the paper is organized as follows: Section 2 presents the methods utilized to build up the literature review. Section 3 presents a discussion on selected literature and data analysis to pinpoint the mechatronic challenges. Section 4 evaluates current solution support and builds up an understanding of important challenges, which are evaluated to be not well addressed. The case study in Section 5 is utilized to present real world mechatronic design scenarios, and to argue about how well they are supported through current solutions. The paper concludes by a discussion in section 6 and a conclusion in section 7.

2 METHOD OF INVESTIGATION

The objective of this paper is to identify the mechatronic challenges, assess their solutions, and illustrate those challenges and solutions through a case study. In order to accomplish this, a literature study is carried out, incorporating contributions from two sources. The first source consists of researchers from the ASME mechatronics community, whereas the second source is based on the collective knowledge of the authors of this article about important contributions within the research of
mechatronic design complemented with a workshop to add on the list of researchers. For the first source, a filter function is needed to sort the vast amount of contributions in which mechatronic challenges are described. The filter function is based on the idea of extracting a large number of references from mechatronics related articles from the ASME conference, and then selecting the most cited researchers. The proposed solutions to the stated challenges are obtained from sources 1 and 2, and from the knowledge of the authors regarding solutions available from the literature. In order to illustrate the findings in terms of challenges and their solutions a case study is used.

3 LITERATURE STUDY

The goal of the first part of the literature study regarding the ASME community is to find the most reported and described challenges. This will be explained in detail in the following sections.

3.1 The Procedure for gathering the data

The three most recent ASME IDETC/CIE conferences are selected for the search, namely the 2008, 2009 and 2010 conferences. The process of finding significant literature is based on identifying researchers who have published mechatronics related articles, and researchers who are cited in the mechatronics community, since both constitute a significant contribution. The aim is therefore to find mechatronics related articles and subsequently extract the references to see who are referenced the most in the community as a proof of relevance. Firstly, articles dealing with the mechatronic design process have to be identified. This is done by using the keyword ‘mechatronics’. If it is ambiguous whether or not the article would describe issues related to the mechatronic design process, the article is read to clarify the content. From the resulting 20 articles, 508 references are extracted.

3.2 Data analysis

The 508 references extracted from the ASME conferences are analyzed by a word-count software to reveal the names that appear the most. This quantitative evaluation is backed up by a qualitative scrutinizing of the reasons why the researchers are ranked as they are. Since it is common that authors cite their own previous work, a precondition is made that an author cannot appear more than once in the reference list of an article. The result of this evaluation is presented below in terms of a name and a numbered code. The first number shows the number of articles in which the researcher has been cited. The second number shows how many times the researcher has been cited in total. The third number is how many articles the researcher has published in the investigated conference proceedings (among the 508 extracted references). G. Pahl (11/12/0), W. Beitz (11/12/0), K.L. Wood (6/13/0), T. Tomiyama (6/13/1), C.J.J. Paredis (5/8/1), R.B. Stone (4/24/1), N.P. Suh (5/6/0), S.W. Szykman (5/6/0), J. Hirtz (4/4/0), D.A. McAdams (4/17/1), T.R. Browning (4/7/0), J.P. Clarkson (4/9/2), J. Gausemeier (4/17/1), U. Frank (4/9/0), U. Lindemann (4/16/2), A. Schmidt (4/11/2), Y. Umeda (4/7/0), M. Yoshioka (4/4/0). Researchers who are cited in less than four articles are omitted from the list, since it is assumed that the above undiscovered list of researchers will cover the needed challenges. Furthermore the number of researchers to consider has to be kept to a manageable level.

The presented search algorithm has limitations. It does not take into account if close colleagues are citing each other, or if the researcher is cited because his/her work is claimed not to be ‘sufficiently good’ by others. Even though the impact of the research might not be directly reflected by the number of citations, the identified researchers are considered to have contributed significantly to the mechatronic community. Therefore their formulation and insight into the challenges faced by the design teams when developing mechatronic products are of importance to this study.

When investigating the researchers and their co-authors, certain research groups appear due to preferred research partners. In the following, the researchers from the list presented above are listed with their preferred research partners.

- Pahl group: Pahl, Beitz.
- Tomiyama group: Tomiyama, Umeda, Yoshioka.
- Gausemeier group: Gausemeier, Frank, Schmidt.

Since researchers within a group tend to have similar views on mechatronics challenges, the grouping simplifies the data analysis.

3.3 Researchers added from the second source

We believe that it is beneficial to extend the systematically generated list with other researchers, who, to our knowledge, have relevant work regarding design of mechatronics. These researchers come from the second source, where the articles are obtained by performing a general search for publications related to design of mechatronics. This search targets the conferences under the design society, relevant journals along with knowledge about mechatronics research groups located at various places around the world. 50 relevant articles from these sources are analyzed and shortlisted based on their significance and relevance towards design of mechatronic products. This provides a list of 19 articles. It is noted that about half of these articles are written by researchers also appearing on the ASME list. Those researchers who are either not cited or who did not publish in ASME proceedings in the last 3 years include Buur [2], Salminen [3], Andreasen [4], and Adamsson [5]. The primary commonality among these researchers is their focus on the conceptual phase of the development life cycle, along with their emphasis on promoting collaboration between designers during the design activity.

The new extended list of researchers was then discussed in a workshop with researchers belonging to the ‘Section of Engineering Design and Product Development’ at the Technical University of Denmark, and the joined list was judged to be comprehensive.
Table 1: Matrix relating mechatronic challenges and researchers stating them. $^1$ = Source 1 researchers, $^2$ = Source 2 researchers

<table>
<thead>
<tr>
<th>Category</th>
<th>#</th>
<th>Challenges</th>
<th>Researchers/Research Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>A</td>
<td>Lack of a common understanding of the overall system design</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Difficulty in assessing consequences of selecting between two alternatives</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Lack of a common language to represent a concept</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Modeling and controlling multiple relations in the product concept</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Being in control of the multiple functional states of the product</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Transfer of models and information between domains (expert groups)</td>
<td>X</td>
</tr>
<tr>
<td>Activity</td>
<td>G</td>
<td>Synchronizing development activities</td>
<td>X</td>
</tr>
<tr>
<td>Mindset</td>
<td>H</td>
<td>Different tradition within the domains for how to conduct creative sessions</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Reluctant to interact with engineers from other disciplines</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>Different mental models of the system, task and design related phenomena</td>
<td>X</td>
</tr>
<tr>
<td>Competence</td>
<td>K</td>
<td>Lack of common language to discuss freely at creative meetings</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>Education within disciplines do not call for integration in professional life</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>The nature of design is different</td>
<td>X</td>
</tr>
<tr>
<td>Organization</td>
<td>N</td>
<td>Product complexity affects the organization complexity</td>
<td>X</td>
</tr>
<tr>
<td>Aspects</td>
<td>O</td>
<td>Knowledge transfer between domains is inadequate</td>
<td>X</td>
</tr>
<tr>
<td>Other</td>
<td>P</td>
<td>Lack of a broadly accepted methodology</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>Mechatronic ownership is lacking</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>System engineers are lacking detailed information of the system</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>Complexity as a generic problem</td>
<td>X</td>
</tr>
</tbody>
</table>

### 3.4 Challenges identified

It is now possible to extract the statements regarding challenges in mechatronic design. For this purpose, between three and five papers from each researcher or each research group are investigated. Based on the extracted statements from each researcher, a KJ [8] equivalent methodology is applied, by which clustering of statements can be performed. A headline for each cluster is then formulated which should embrace the statements clustered in it. In Table 1, these headlines for the challenges are listed. The highlighted rows in the table are used for illustration of points discussed in section 4 and, thus will not be discussed in this section. In Table 1, the link to the researchers whose work complies with the stated challenges is also marked. The stated challenges are causally linked. As an example, the ‘lack of common methodology’ leads to a ‘lack of a common representation of a product concept’. However, the causal chains will not be discussed further in this article.

Table 1 cannot be assessed quantitatively since the pool of data, being the number of researchers investigated, is relatively small and because the filtering process has distorted the picture of how many times a specific challenge is mentioned. The distortion occurs because it was chosen to group some of the
researchers, which affected the number of times, the challenges appeared.

In this paper, for the sake of simplicity, all the identified challenges in Table 1 are assumed to be generic and hence, not context specific. This is done regardless of the number of times these challenges are repeated by the researchers.

In Table 1, some of the researchers stand out. These are Pahl and Beitz, the Wood group and Suh. Their work is often cited due to their fundamental contribution to design theory, and even though they address mechatronic or complexity issues in their work, a large part of the mechatronic-specific challenges in Table 1 are unaddressed.

What have we gained by introducing the researchers from source 2? The conclusion falls in two parts. Firstly: The fact that a large part of the challenges is also stated by the researchers from source 2 supports the claim that the challenges stated by source 1 are truly generic and thus important to direct attention to in research. Secondly it can be observed that some of the challenges are only described by researchers from source 2. Even though they are not validated to the same extent as those described by both sources, they still add to the understanding of the multidimensional challenges experienced by design teams developing mechatronic products. The researchers contributing with new challenges are Adamsson, Buur, Andreasen, and Salminen.

The most commonly reported sets of challenges are primarily related to the way a product concept can be described and how information linked to the product concept can be shared across engineering disciplines. The commonly observed challenges are (the highlighted rows in Table 1 is not linked to this list): ‘A Lack of common understanding of the overall system’, ‘A lack of a common language to represent concept’, and ‘A lack of a common language to discuss freely’. As stated by many of the researchers, the fundamental reason leading to the many challenges is the absence of a common mechatronic design methodology. This is again rooted in the fact that theories building upon different axioms cannot be joined to a common theory, as described by Tomiyama [1].

4 SOLUTIONS PROPOSED

This section will present a number of solutions to the mechatronic challenges, which are compiled through the literature study presented in Section 3.4. When there is sufficient documented evidence that a certain proposal addresses one or several challenges in design of mechatronics, we consider it as a solution as listed in Table 1. The table shows the challenges, which a given solution aims to support. The primary focus of a solution is represented by an orange cell. A ‘Y’ marks that a challenge is sufficiently addressed by a solution, whereas ‘P’ indicates that it is partially addressed. The process of allocating the ‘Y’ and the ‘P’ was carried out by the authors of this paper who are active researchers within the area of mechatronics.

A general overview of Table 2 shows that the mechatronics challenges are not sufficiently addressed by the proposed solutions. Specifically, solutions for challenges B, C, F, G, K, M, N, O, P, R, and S are either partially defined, or no solution is proposed. Among these challenges, there are challenges that relate to 1) competences (K, M), 2) to activities (G), 3) to organizational level (N, O). These are not treated further in this paper, since our scope is not towards competences of individuals in a company, nor the synchronization of activities or the organizational issues. Challenges B, C, and F, and challenges P, R, and S are strongly connected to each other because of the following reasons: Since there is a lack of broadly accepted methodology (P) in mechatronics, a common language to represent the concepts can be difficult to accomplish. This creates a problem of finding the most suitable design through efforts across different domains. Along with difficulty in assessing consequences, the lack of methodology and the lack of a common language contribute to a higher complexity (S) in mechatronics. In addition to that, the lack of common language and inadequate information transfer between domains are strongly connected to challenge R (system engineer lacking detailed information on the system). Therefore, to gain a detailed insight on some of the core challenges in mechatronics, we will restrict ourselves to challenges B, C and F, which we believe are at the heart of mechatronic challenges. B, C and F are marked in green in Table 1. The other challenges are also important, but not treated further to limit the scope of this paper.

In the following, each solution is discussed and assessed about how well it supports challenges B, C and F.

1) The first solution from Table 2 is about methods based on functional thinking. Buur [2], Wood [9], Tomiyama [1], and Suh [10] are examples of functional approaches. Functional modeling is abstract in terms of the level at which the description of the product concept is performed. Therefore it can serve the purpose of a common modeling language (C) to an abstract level only. It is typically after the functional modeling that the development process becomes domain-specific. Functional thinking is only part of the complete picture of the design activities, and other factors such as structural consequences and effects of system elements onto various system properties is not supported through it (B). Moreover, since models contain much more information than functions; functional thinking is only useful when transferring information between two abstract system models (F).

2) The second solution is about modeling relationships between elements from different mechatronic domains. Design structure matrix (DSM) and domain mapping matrix (DMM) by Lindemann [11], [12], and Browning [13] are examples of modeling relationships between functions, components, physical structure, and resources during a mechatronic product development process. Bonnema [14] proposes a FunKey (function keydrivers) architecting approach to model relationships between functions and key drivers of a product, in an aim to provide good insight to different stakeholders while designing. The main aim of these approaches is to support the managing of multiple relations and dependencies during design. However,
analyzing the consequences of selecting between different alternatives becomes too cumbersome through these approaches in terms of effort and efficiency (B). Moreover, they are meant for establishing relations across different domains, and not for information transfer between design models (F). They are partially suitable for aiding a common language (C), since engineers from different domains can discuss dependencies based upon them.

3) The third solution is about controlling integration between domains via requirements. Systems Engineering [15] and work by Tomiyama [16] are examples of such solutions. However, requirements cannot be utilized for accessing consequences (B) of different design alternatives for a mechatronic system. Therefore, model-based system engineering [17] proposes to utilize requirement management tools in addition to system-level modeling (common modeling language (C)) to control system design based on requirements. This provides a better utilization of requirements through a computer support. A model transformation between system-level models and domain specific models (F) is however required to keep the design models consistent with each other.

Table 2: Solutions proposed in literature against the challenges identified in Section 3.4. An orange cell indicates the primary aim of the solution, and the green columns show the most important challenges

| #  | Solutions                                                                 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| 1  | Activities based on functional thinking (Buur [2], Wood [9], Tomiyama [1], Suh [10]), applying functions, means patterns, state and event relations (process related) | P | Y | Y | Y |   |   |   |   | P | Y |   |   |   |   |   |   |   |   |   |   |
| 2  | Relationship management e.g. DSM and DMM [11], [13], QFD [18], FunKey [14] |   | P | Y | P | P | P | P |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3  | Controlling design activities through requirements management (Systems engineering [15], Tomiyama [1]) | P | P | P | P | P |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4  | A process model containing activities for the development process. (Isermann [19], VDI2206 [20], Salminen [3], Systems Engineering [15]) |   |   | P |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5  | Informal description consisting of a number of modeled/described aspects to specify systems, A3 overviews [21], Salminen [3], Buur [2] | Y | P | P | P | Y | P | P | P |   |   |   |   |   |   |   |   |   |   |   |   |
| 6  | Modeling languages to describe system as a whole, formally or semi-formally. SysML [22], SPSL by Gausemeier [23] | Y | P | P | P | Y | P | P | P |   |   |   |   |   |   |   |   |   |   |   |   |
| 7  | Model transformation from a design model in one domain into a design model in another domain (Gausemeier [24], Paredis [25]) |   |   |   |   |   | P |   |   | P | P | P | P | P | P | P |   |   |   |   |   |
| 9  | Simulation of phenomena that incorporates elements from the different domains (e.g. Modelica [27]) | P | P | P | P | P | P | P | P | P | P |   |   |   |   |   |   |   |   |   |   |
| 10 | Setting up a systems integration group in the project (Adamsson [5], Andreasen [4]) | Y | P | P | Y | Y | Y | P | P | P | P | Y | P |   |   |   |   |   |   |   |   |

4) Different process models, specifying the activities to be performed during the design process are proposed by several researchers. These process models are usually an extension of a process model in one domain towards covering several domains. VDI2206 [20], Systems Engineering [15], and work by Isermann [19] and Salminen [3] are examples of such models. These models aim at synchronizing the workflow and activities, which the design team must perform during the development. However, these approaches state that dependencies should be handled, not how to actually manage them in relation to assessing consequences (B). A process model urges to utilize a system design language such as SysML for systems engineering. However, process models themselves do not solve the common language challenge (C). The same applies to model transformations (F), which can be made a part of the process models, but process models themselves do not aim at solving challenge F.

5) In the aim of a common language and to solve the communication problems during conceptual design phase, different solutions are proposed (solutions 5, 6). An example of an informal description is the A3 architecture overviews [21], which provides an overview of the complete system, in terms of different system aspects, such
as functional aspects, physical aspects, etc. Although representing system design concepts informally is useful to discuss system design among different domain experts, however, assessing consequences for each design domain while choosing different system alternatives is not addressed (B). Moreover, such overviews have the same potential of becoming a common language as the functional thinking proposed by Buur [2]. Hence, there are other abstraction levels in design that cannot be supported by A3 overviews (C). Presenting concepts in A3 overviews can lead to gaps between domain-specific design activities and system-level design activities, and it is clear that model transformations cannot be utilized with A3 overviews to reduce this gap (F).

6) An attempt towards a language more specifically related to mechatronics is the semi-formal specification language (SFSL) by Gausemeier [24], which aims at specifying a mechatronic concept in terms of a number of aspects, such as a behavior-aspect and a structural-aspect. Modeling languages having formal semantics that describe the system in terms of different views are also proposed such as SysML [22]. The opinions from researchers behind these modeling language approaches contain a contradiction, especially in terms of their usefulness and effectiveness. For example Borches et al. [28] document that formal modeling such as SysML does not usually solve the communication problem between people from different design domains, nor does it produce models that are easy to understand. The fragmentation of proposals for a common modeling language by different groups of researchers indicates a need for further improvement in this area. Therefore it can be said that although a common design language is a need, the nature of such a language in terms of being formal or informal is still unknown, and there is still a need for developing support in this area.

7) Model transformations are proposed as a possible solution to relate two design models. Shah et al. [25] shows how a mapping between two design models can be used to build transformations between them. An example is the transformation between SysML [22] and Modelica [27], which combine the descriptive capability of SysML with analysis and simulation capability of Modelica. Formal models utilized during conceptual design phase have advantages of supporting automated-model transformations to other design models. However, dependencies between mechatronic domains cannot be directly solved through model transformations, because it is not always possible that a model contains a representation of all possible dependencies that arise while accessing consequences of different alternative design solutions (B). The dependencies that are important and the consequences that are critical to be considered are not necessarily known beforehand (the uncertainty element). Moreover, model transformations (F) can be more effective if a proposal for a common design language (C) in mechatronics becomes successful. However this is not an explicit goal of the model transformation community to develop such a language.

8) Besides intra-domain interfaces, interfaces can also be observed between domains, such as a shielding of an electronic sensor. An international standard exists (ISO/IEC 81346) that specifies how to define a physical interface. Furthermore, clearly defined interfaces are stated as being advantageous [15]. The interface description aims at specifying the physical interfaces based on a functional partitioning between the domains. Therefore, interface handling can only provide some of the information needed for assessing consequences. Hence, B is not covered. Clearly stated interfaces cannot be used as a common language (C), even though it can be used as a framework for discussions. Model transformation (F) is decoupled from interface specification, and is therefore not covered.

9) Computer aided multi-domain modeling and simulation provides advantages of building design models with elements from different domains, along with executing them in order to assess certain product properties. These modeling languages are well supported by tools, in order to conduct simulation and analysis. Modelica is one example of such modeling languages. Although such languages provide support for assessing consequences (B) to an extent, they cannot be treated as a common design language (C) for all domain experts. Moreover, they are only good for design modeling when the basic principles and the basic structure of the product have been determined. Current efforts within the Modelica community aim to standardize model transformations (F) between SysML and Modelica. However this will only be useful if SysML is utilized.

10) Adamsson [5], and Andreasen [4] proposed setting up a systems integration group. This group is primarily responsible for facilitating the information flow, and the collaboration between engineers from the different domains to increase performance of the overall system. However, challenges B, C, and F are only supported indirectly by anticipating that an integration group will facilitate closer integration between the domains.

5 CASE STUDY

The purpose of presenting a case study in this article is to illustrate the three selected mechatronic challenges (B, C, and F) highlighted in section 4. This will allow us to relate the rather abstractly described challenges to a very concrete situation in a product development process. Additionally, the product case will help assessing how well the proposed solutions would have helped the design team in their design task. Therefore, the case study is not used for verifying whether or not a challenge can be handled by the proposed solutions. Instead, it is used to bring in a real-world dimension, and create a context surrounding the challenges, and thereby understand better what it might take to create satisfactory solutions.

The aim of the project, chosen as the case study, was to develop a watch system based on the idea to develop a mechanical watch and an instrument, which can be attached to
the watch (see Figure 1). The instrument contains advanced functions used for alpine skiing. An additional two external units wirelessly transmit heart rate information and temperature information to the instrument attached to the watch. In this case study, we focus on the external temperature unit shown in Figure 2. It is noteworthy that one of the authors was involved as a development engineer in this specific project.

![Figure 1: 1. Instrument, 2. Watch, 3. Temperature Unit, 4. Heart-rate sensor, 5. Charger for instrument](image)

The case study was built up on the experience gained by participating in the development team backed up by document-analysis, and interviews with the project managers for the mechanical, electronics and software development. Due to limitations of describing the development process as a whole, we deem it necessary to only select small fragments from the design process to illustrate the selected challenges. In the following, three scenarios from the case study are presented, which are directly related to what we consider as the most important challenges (B, C, and F). This is followed by a discussion on possible solutions from Table 2, and a conclusion on using those solutions to mitigate the particular challenge.

![Figure 2: Temperature Unit](image)

### 5.1 Assessing consequences (Challenge B)

#### The Power consumption scenario: In the beginning of the project, it was assessed that the power consumption would be one of the major key drivers for the project. The RF chip for wireless transmission and a running processor are the primary sources for the power drainage. The main electronic components are illustrated in Figure 3. Two basic approaches can be chosen: either to minimize the power consumption (thereby the user should change batteries), or to make the whole unit rechargeable. Within the scope of minimizing the power consumption, two main directions can be chosen, which is either to cut the power manually or automatically when it is not in use, or to minimize power usage by features in the electronics, and by clever programming. Solutions are spread over all the domains. Some solutions have a direct effect on the use pattern, hence the user experience. Some solutions require further technology clarifications. Other solutions require the consequences on the products life phases (e.g. change of battery) to be assessed. The main challenge is that there are many conceptually different ways of solving the power issue, but how can we, in the best possible way, reason about the consequences of selecting one product concept above another? The problem of assessing the consequences when choosing between concepts is a general concern in product development. Yet, this concern increases when different domains are involved in the design process while investigating alternative design concepts.

#### Discussion on solutions: In Table 2 four solutions have been identified, which potentially should embrace the challenge of assessing the consequences by selecting between two product concepts: a) Relationship management; b) Informal descriptions; c) Formal language description; d) Mechatronic concept description and simulation of phenomena. DSM, MDM, QFD as well as formal modeling languages such as SysML and the various simulation programs only provide a description of a single or few closely related properties or aspects. In the case study, a holistic approach is needed to consider the consequences of a product concept, which the mentioned mechatronic solutions cannot encompass. In the project, various concepts were sketched to reveal their potentials and drawbacks and to evaluate the life phases. The product concepts were then discussed on several meetings and the progression of reducing the needed power was continuously assessed.

Modeling languages exist ranging from the formal modeling languages such as SysML over semi-formal modeling languages such as Guausemeier’s SFSL, to less restricted modeling such as the A3 overviews. Even the A3 overviews, which is proposed as an informal method, is not sufficient, since it does not address mechatronic specific aspects such as the implications of different allocation of functions to the domains. An informal description different from the A3 overviews, seems to be the best way to mitigate the challenge since an informal description is flexible. The question is, however, is the informal description so flexible that it does not provide any mechatronic specific support? The answer seems to be yes. In the presented case, the solutions from Table 2 seem even less appropriate than evaluated in the table to solve the problem of assessing the consequences by selecting between two or more product concepts.

![Figure 3: Main Electronic Components](image)

### 5.2 Common language to represent a concept (Challenge C)

#### ‘The custom made gasket’ scenario: A request for changing the outer shape to make the unit appear lighter causes a change of the mechanical design (Figure 4). The changed
design makes less space for fitting the main gasket, which ensures the water resistant property of the unit. Instead of the previous used standard O-ring, a custom shaped gasket must be used unless the outline of the PCB is changed (Figure 4). At this late stage of the electronic development, a change of the PCB would result in reorganizing the electronic components. In a HF circuit, the relative placement of components affects the transmission quality, thereby increasing development cost, risk and time if the PCB layout were to be redesigned. Therefore a custom made gasket is evaluated as ‘the best fit’ solution.

Figure 4: The custom made gasket and the part in which it has to be inserted

Discussion on solutions: The situation as described above is a known characteristic of the design of mechatronics, where the best alternative among few has to be chosen, such as changing the gasket or changing the PCB. However, there are consequences attached to each alternative for different design domains, such as the redesign cost of the PCB, the redesign cost of the gasket and the mechanical module, the packaging of the high-frequency electronics, and the success probability of the integration test. The dependencies between different domains during the design activity are major contributors towards these consequences. For example, the relation between the gasket and the size of the PCB. Moreover, the best solution has to be considered in terms of the overall system, and not just between domains. Considering Table 2, the common modeling language proposals such as SysML, SFSL, and A3 overviews can be considered to build up a system view, hence supporting modeling and evaluation of alternatives in terms of the system as a whole. Moreover, DSM/DMM, and FunKey architecting are also proposals to identify relations between functions and user demands, and between functions and components. However, DSM/DMM and FunKey architecting serves the purpose of relationship management only. Building a holistic system view along with assessing certain characteristics of the system such as performance or cost is not supported. From Table 2, activities based on functional thinking, and controlling design through requirements are also proposed as solutions for a common language to describe the concept. However, functional thinking is proposed to describe only the functional view of the product, thereby leaving out the structural view which is essential to the gasket issue. In the case of requirements, they can be used for goal specifications (of the product to be), or result specifications (documenting the finalized product), but requirements cannot be used to represent a design concept.

Considering SysML, SFSL, and A3 overviews, these languages provide different solutions towards representing the size constraint relation between the gasket and the PCB. This constraint modeling enables mechanical and electrical/electronic engineers to understand the effects of gasket size on the PCB. It also relates this constraint to the complete system model. However, the decision for whether to redesign the gasket, or redesign the PCB requires assessing consequences of each alternative in relation to designer preferences. We believe that availability of an informal and visual language, where designers from different domains can sketch their ideas to each other, and highlight the relationship of their concepts to each other, is a more effective way of managing dependencies such as between gasket and PCB. The sketching can be partially or fully supported by a calculation or a simulation engine (depending upon how open/restrictive the visualization is). SysML, SFSL, and A3 overviews are common in certain aspects; however, they differ in terms of being formal, semi-formal or informal. Especially A3-overviews is an informal medium to discuss such dependencies between views. However, it does not target how can these dependencies be understood and managed during the design activity. It rather defines a medium where these dependencies can be expressed in a way that is understandable to different engineers. The usefulness of formal or semi-formal modeling (such as SysML) is explained to be not useful in the conceptual design phase due to the rate at which models change, and due to decreased communication effectiveness caused by a lack of visual representation of the structure of the product by different engineers [28].

5.3 Transfer of models (Challenge F)

‘The ESD protection issue’ scenario: Due to a requirement for better temperature sensing, a change of the design is necessary. Discussing the proposed solution with the electronic engineers, it becomes apparent that this type of solution is prone to electrostatic discharges and that mitigations have to be made for the electronics not to be damaged in such a case. The proposed design is shown in Figure 5.

Figure 5: The PCB and the positioning of the flex print and the flex print terminal

For easier handling of the small thermo sensor, it is placed on a flex print which can easily be connected to the PCB compared to five ordinary wires. Due to the stiffness of the flex print, the location and orientation of the terminal is important and this fitting is made in corporation between the electronics
and the mechanical engineer. Figure 5 shows the PCB connector placement and the position of the flex print.

**Discussion on solutions:** In this particular case, the orientation and location of the terminal on the PCB, and the placement of the connector on the flex print is a clear dependency between electronics and mechanical models. In order to reach a solution, both electronics and mechanical engineers had to have several discussions during a number of design iterations. From Table 2, three solutions have been identified which should aid in overcoming this challenge related to information transfer across domains: a) Controlling the design through requirements management; b) Simulation of phenomena incorporating model elements from different domains; c) Integration of models through model transformations.

Requirement specifications play a key role in controlling the design, and hence it is proposed to utilize these specifications as a solution to ease the information transfer between domains. Traditionally, a specification has to direct the search for solutions. What is required (here) for information transfer is the detailed information/representation of needed parameters of a concept from each domain, and not the specification that directs search for those concepts. Simulation of properties is also proposed as a possible solution to ease the information transfer between domains. However, in the above case, the mechanical and electronics engineer need information regarding the location of the terminal. Hence, simulation in the sense of algorithmic optimization cannot be utilized for this task.

Integration of models through a model transformation such as [25] and [24] is proposed as a solution to aid in information transfer between domains. The location of the connector in the mechanical design model can be extracted and represented through a transformed model i.e. an electronic design model to facilitate the electronic engineer during the design process and vice versa. In the following, the relationships between models are more closely described to be able to evaluate challenge F in terms of performing a model transformation.

Different design models are related in terms of system properties which they affect. Although two design models may both affect one system property, there is only a portion of each model that has substantial meaning in the other model. Tomiyama et al. [29] explain that two models can only be integrated with each other if the background theories (that these models are based on) are compatible. The compatibility between two background theories suggests that a concept in one theory can be related to a concept in another theory. For example, inertia has no meaning in electronic PCB design, but has a meaning in controller design. If two background theories are compatible, then a model transformation can be applied to the corresponding models. Model transformation approaches provide a capacity to control which part of the source model is read and what is created in the target model by specifying meta-models and the transformation between them. Therefore, we conclude that model transformation has a potential in addressing challenge F. In the following, model transformation approaches will be discussed further, followed by concluding remarks on the limitations of a model transformation.

One approach for integration of models is to utilize a central product model where all the information is stored. The central product model can be utilized to understand and manage the relationships between different aspect models. The aspect models can also be generated from the central product model. Another approach for managing relationships between models is where an integration at the level of background theories is proposed to support integration between so called ‘multiple aspect models’ [29], [30], and [31]. The approach is based on developing different aspect models based on different background theories, e.g. dynamic models, and geometric models. These aspect models can be integrated through a central meta-model, where the relationships between the concepts of the different background theories are specified. Specifying the concepts and the concept relationships between the different background theories in a meta-model aids in managing the influence of a model element in one aspect model onto a model element in another aspect model. A similar approach is presented in the PACT experiment in [32], where an approach for integration among multiple aspects (agents) during design is discussed.

It is likely that a transformed model does not contain all the information that is required by a modeler, because it is not always possible to know at earliest stage which properties affect each other and hence, should be in the model. This information might be known at a later stage, and if these properties are not explicitly supported by the meta-model of a domain, then a model transformation will not be useful straight away, and will require further efforts. Hence challenge F is not fully addressed through model transformations.

In order to support the design process for mechatronic products, we propose model integration between domain-specific views such as a mechanical view and a system view built through a common system modeling language. This will provide an opportunity to find a best mechatronic design solution for a system. [25] and [24] are examples of steps in this direction. However since the nature of common modeling language is still an unknown, this area has a good potential for further development.

6 **DISCUSSION**

Most papers about mechatronic design end by stating that a common methodology and a common conceptual model is needed. This statement has been repeated for the last 20 years. If it was possible, it would have been likely that such a method would have been found, or significant findings presented which would be a step towards it. Proposals of a mechatronic concept description always end up by constituting different needed views. Having ‘x’ number of different views on a concept negates the idea of a common conceptual representation. In principle, this is not different from the acknowledgement that you need several different views of a system to be able to describe it, e.g. the proposal of the domain theory by Mogens Myrup Andersen in the early 80’ies [33], also described in
Tomiyama [1] states that two theories cannot be joined if they are based on different set of axioms. This is the reason why the so-called ‘common mechatronic concepts’ always have to be presented by ‘x’ number of views. For each type of property one has to model, a separate view has to be created [35], [30]. However, there are some theories that by nature share the same axioms across the domains. Jacob Buur [2] pointed out the functional thinking as a common theory that can be applied to all domains. This will enable methods, which are based on functional thinking to be used across the domains in mechatronic development. Some of the methods based on functional thinking are: life phase thinking, process descriptions of the product, state-transitions, function/means tree, and QFD. Quite soon in the development process, one needs to model and evaluate properties of the design. Whether the property modeling is performed based on sketching and/or computer simulation, the problem of a common mechatronic model appears, because an evaluation of a property is linked to a certain theory which will be domain specific. To assess several properties from different domains in one model, no adequate theory or tool or process has been proposed. We suggest the following thinking experiment: If two competing concepts are developed to finalized products, the consequences can be fully evaluated. Since this is seldom carried out for obvious reasons, it is necessary to show the relations and consequences by other means. We have previously described that a common conceptual model, which has details beyond describing functionality in the product, would violate the fundamental axioms. Therefore, we have to accept that not all the relations can be modeled, besides those few which can be described as the key relations. We should be willing to work with ill-defined problems across the domains and willing to generate alternatives and most of all to be able to identify what information is relevant to share with developers from other domains. We should acknowledge the ‘collaboration’ research field (the human aspect), and provide room (workshops) and methods, which will enable cross-domain discussions, and which will be graphically intriguing.

7 CONCLUSION

In this paper, the challenges, which seem to be most significant in the design of mechatronics are presented. The proposed solutions in the literature only provide partial solutions to those challenges. A large part of the identified solutions appear to support analysis rather than synthesis. As a product concept progresses, effort must be spent to continuously update the information that goes into the analysis-oriented solutions to be able to use them. This effort compared to what can be gained by using a particular solution is seldom assessed, evaluated or investigated in the literature. The solutions which are not analytical in nature are the ones based on functional reasoning, which have the capability of being applied across domains. Unfortunately, these solutions are not well described in terms of how to apply them to an actual synthesis process of a mechatronic product. Even though functional reasoning should be capable of supporting the design process through all the design phases, the suggested solutions only support the initial steps in the conceptual phase.

A common design language would, as stated by many of the researchers in the study, facilitate a better collaboration between engineering disciplines. A common language, if possible to develop, would need to consist of ‘x’ number of product views to be modeled, ruling out the prospect of a unified representation. Furthermore, a common language should be evaluated based on: its capability to represent the desired views effectively, its potential to be understood by engineers from the various domains, and its effect on the efficiency of the development process. If a common language can be realized, it would also facilitate in creating variations of the product concepts in the conceptual phase. The case study illustrated this as being beneficial to reveal the consequences of selecting between alternative design concepts.

ACKNOWLEDGMENTS

We wish to thank Morten Linde and Jorn Werdelin, who are the co-founders of the watch company LindeWerdelin [36] for letting us use the development project as the case for this article. We would also like to thank Troels Pedersen from IPU and Henrik Bendsen from Oticon for their help in adding detailed information to build the case.

REFERENCES

2009.


