COMBOS: Communicating Behavior Of Systems

Incorporating simulations in conceptual system design

Steven Haveman

COMBOS: COMMUNICATING BEHAVIOR OF SYSTEMS

INCORPORATING SIMULATIONS IN CONCEPTUAL SYSTEM DESIGN

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 3 december 2015 om 14:45 uur

door

Steven Pieter Haveman

geboren op 1 augustus 1986

te Voorburg

Dit proefschrift is goedgekeurd door:

Prof. dr. ir. F.J.A.M. van Houten	Promotor
Dr. ir. G.M. Bonnema	Assistent Promotor

COMBOS: COMMUNICATING BEHAVIOR OF SYSTEMS

INCORPORATING SIMULATIONS IN CONCEPTUAL SYSTEM DESIGN

PhD Thesis

By Steven P. Haveman at the Faculty of Engineering Technology (CTW) of the University of Twente, Enschede, The Netherlands

Enschede, 3 December 2014

Dissertation Committee:

Prof. dr. G.P.M.R. Dewulf	University of Twente (Chairman and Secretary)
Prof. dr. ir. F.J.A.M. van Houten	University of Twente (Promotor)
Dr. ir. G.M. Bonnema	University of Twente (Assistant-Promotor)
Prof. dr. G.J. Muller	Buskerud and Vestfold University College, Norway Embedded Systems Innovation by TNO, Eindhoven
Prof. dr. J. Hooman	Radboud University, Nijmegen Embedded Systems Innovation by TNO, Eindhoven
Prof. dr. ir. B.R.H.M. Haverkort	University of Twente
Prof. dr. ir. J.I.M. Halman	University of Twente
Prof. dr. ir. S. Stramigioli	University of Twente

UNIVERSITY OF TWENTE. COMMIT/

This research was supported by the Dutch national program COMMIT and carried out as part of the Allegio project.

ISBN: 978-90-365-3971-5
DOI: 10.3990/1.9789036539715

Printed by Ipskamp Drukkers, Enschede

"We have two ears and one mouth, so we should listen more than we say."

– Zeno of Citium

Preface

Well, there it is! The result of my research during the past four years. After a successful defense of the thesis, I can even call myself PhD, or doctor in Dutch. A part of my mind tells me that this is an achievement to be proud of. However, the other part tells me this is a logical outcome of a process that I started. I guess this has much to do with your environment (and of course your mindset). During the research, I was working with other PhD candidates, in touch with other researchers having done a PhD or completing a PhD and even in my personal life I started noticing more and more PhDs. While sometimes people marveled at the complex things I was supposed to be doing, I merely shrugged my shoulders, because to me it was all pretty logical. Nevertheless, I do recognize that some things have changed. This is especially noticeable when rereading literature for a second time after a few years. Suddenly, you are able to read between the lines and often I have asked myself: "why did I not realize this before?" Thus, I can happily say that I learned a lot!

The seed for pursuing a PhD was planted when I found myself surrounded by PhDs during my Masters' graduation assignment. The opportunity to dive into a single subject for four years and emerge as an expert in that field appealed to me very much. Especially because I felt that this specialization was lacking so far in my education. After my Master, this ambition was not fulfilled immediately. However, after working in industry for two years, an opportunity arose to return to the University of Twente. I am happy that I got the position in the end. Therefore, I would like to thank both my promotor, Fred van Houten, and especially my assistant promotor, Maarten Bonnema, for this opportunity. Maarten, your input and constant questioning of my research have been much appreciated. At times it might have been testing as the process of conceptualizing a tangible approach from my research proved to be difficult. I thank you for your confidence in me and your consistent and honest approach of my supervision.

Other acknowledgements with regards to my research have to be made to my colleagues in the Allegio project at Philips Healthcare. Dave, Arjan and Jozef, thank you for your guidance. Jozef, you are also thanked for being part of my dissertation committee and for our talks during the afternoon train rides. Nicolas, Sarmen and Freek, it was a pleasure to work with you. I would also like to thank various people at Philips iXR that I have collaborated with during my time there. Gernot Eggen, Rob Albers, Robert Huis in 't Veld, Pascal Wolkotte, Mathijs Visser, Mathijs Schuts and Laura Calvino, it was interesting and worthwhile to collaborate with you. Finally, I thank the interviewees at both Philips Healthcare and other companies for their time and input.

I would also like to thank my OPM-colleagues at the University of Twente for providing a fun atmosphere each time I was in Twente. Initially, I felt a bit of frustration at the fact

that only being present for two days a week forced me to miss out on many social activities or intricate schemes that were being developed in N211. However, in the end I managed to tag along plenty of times and got to know a lot of you in more depth. Special thanks go out to everyone who shared knowledge on systems engineering in the System Design Meeting. Last but not least, thanks to everyone who has shared N211 with me over the years, it has been fun!

Every week when I visited Enschede, I stayed the night. First of all, to reduce some of my travel time, but second and more importantly, to visit some of the important persons in my life. During the first years, I stayed with my grandmother. I will always remember those visits fondly. When my parents moved to Glanerbrug in September 2012, I started to alternate between the two locations. Once illness confined my grandmother to her bed in August 2013, we still visited every Monday night, until she passed away in January 2015. However, she remained cheerful until the end and reached the age of 97, so that is something admirable and nothing to be sad about. Pap and Mam, thanks for all the support and I will miss the frequent visits!

Finally, lieve Paulien, you have been the most important person in my life for the last decade. To be exact, 10 years and 1 day once I have defended my thesis. For the last four years you had to spend a night alone weekly, while I was out and about, yet you did that without ever complaining. I hope that we are able to spend many more decades together, in which we can enjoy each other's company. Let's discover the world together during our upcoming journey and hopefully with many adventures afterwards.

Utrecht, 6-11-2015
Steven Haveman

Summary

Understanding the behavior of a system during its design is currently one of the greatest challenges in systems engineering. These systems, for example interventional X-ray scanners, cars or spacecraft, can operate in many different modes, in a plethora of environments and with various goals. Simply said, the behavior of a system has become increasingly complex. Obtaining insight in what kind of behavior is desired and which behavior should be avoided is an important part of the development process. This is done by employing tools such as models and simulations that are able to characterize the behavior of a system. However, this is only one piece of the puzzle. As tools are used to gain insight in the system's behavior, the next step is to share and discuss these insights across a large group of stakeholders from multiple disciplines. This is why supporting communication of system behavior in an effective manner is crucial.

The research presented in this thesis emphasizes the need for this support, explores how this support can be realized in two real-life case studies, presents the COMBOS method to achieve this consistently and evaluates the COMBOS method through examples and discussion.

In the initial, conceptual stages of system design the goal is to define a system architecture that stakeholders understand, support and have confidence in. This architecture should not only be detailed using a structural perspective, but using a behavioral perspective as well. In this thesis, it is argued that a system architecture in conceptual system design should consist of an overall system view that details subsystems and key drivers, as well as specific views for these subsystems and key drivers. The essential views to be represented are the physical, functional, quantification and operational view. However, the conclusion of this thesis identifies that a context view is also essential.

To gain insight in system behavior, various modeling and simulation techniques can be employed. This research has identified four challenges that impede application of simulations in conceptual system design. These are to accommodate multidisciplinary views, to support problem-focused design space exploration, to cope with uncertainty and to deal with a lack of formality. This research offers a framework to address these challenges.

In order to explore the use and communication of simulations in practice, two case studies have been executed at the interventional X-ray division of Philips Healthcare. The first case study explores hand-eye coordination, while the second case study considers the power management subsystem. The case studies show that adequate tools and techniques are available to model and simulate these issues, for example using the Y-chart methodology and the POOSL modeling language. However, the case studies also showed that adequate support to share and discuss the insight gained is lacking. In the first case study, an A3 Architecture Overview is used to communicate the simulation results. While this shows promise, it is also noted that simulations and models generate too much data to capture the required insight in one static A3 sized overview. In the second case study, the system

behavior is visualized over various states in various views. These views can be explored in an interactive system overview by controlling the underlying simulation that is embedded within this overview.

Based on these case studies, this thesis presents the COMBOS (COMmunicating Behavior Of Systems) method. COMBOS embeds behavioral analyses and especially multidisciplinary communication of these analyses firmly into the system development process. The method stresses the importance of continuous communication. The final result of applying the method is an interactive system overview that allows stakeholders to observe the system's behavior in different views simultaneously. In turn, this enables multidisciplinary design discussions that for example focus on changing behavior in the functional view based on possibilities observed due to specific behavior in the physical view. A set of eleven guidelines is offered to support application of the method.

COMBOS has been applied on two example cases, one concerning a subsystem of an electric car and the other concerning a key driver of a space mission. These examples show that application of the method is feasible.

Evaluation of the method shows that COMBOS provides an effective approach to communicate system behavior in the conceptual stages of system design, to support the systems architecting process. This is confirmed through a qualitative survey with involved stakeholders. However, these stakeholders are currently unable to judge the efficiency of the method.

The main contribution of the research presented in this thesis is the fact that the COMBOS method improves conceptual system design by describing how to embed behavioral analysis and especially its communication into conceptual system design. Applying the method will improve conceptual design of systems, because system behavior will be understood and specified better across multiple disciplines. The expectation is that improving the conceptual system design will reduce the discovery of costly errors in later design stages. This ultimately results in systems that have been developed both cheaper and faster and above all, meet the needs of customers and users.

Samenvatting

Het begrijpen van het gedrag van een systeem tijdens het ontwikkelproces is op dit moment een van de grootste uitdagingen op het gebied van systems engineering. Dit soort systemen, bijvoorbeeld interventionele röntgenscanners, auto's of ruimtevaartschepen kunnen op veel verschillende manieren functioneren, in een veelheid aan omgevingen en met variërende doelen. Simpel gezegd is het gedrag van systemen veel complexer geworden. Het verkrijgen van inzicht in het gewenste en ongewenste gedrag is een belangrijk aspect van het ontwikkelproces. Dit is echter maar één stukje van de puzzel, de volgende stap is namelijk het verkregen inzicht communiceren met een grote groep stakeholders uit verschillende disciplines. Daarom is het ondersteunen van de communicatie over systeemgedrag cruciaal.

Dit proefschrift benadrukt de noodzaak voor deze ondersteuning, onderzoekt hoe deze ondersteuning gerealiseerd kan worden in twee case studies, presenteert de COMBOS methode om dit consistent te realiseren en evalueert de COMBOS methode met voorbeelden en een discussie.

In de initiële, conceptuele fases van systeemontwerpen is het doel om een systeem architectuur te ontwikkelen die stakeholders begrijpen, kunnen uitdragen en waar zij vertrouwen in hebben. Deze architectuur moet niet alleen bestaan uit een structuurweergave, maar moet ook aandacht hebben voor het gedrag van een systeem. In dit proefschrift wordt gesteld dat een systeem architectuur in conceptueel systeem ontwerpen zou moeten bestaan uit een algeheel systeemoverzicht dat ook subsystemen en key drivers identificeert. Deze subsystemen en key drivers hebben elk hun eigen overzicht. Essentiële onderdelen van deze overzichten zijn een fysiek, functioneel, kwantitatief en operationeel overzicht. Dit proefschrift concludeert echter dat een overzicht van de context ook essentieel is.

Om inzicht te verkrijgen in het gedrag van systemen kunnen verschillende modelleer- en simulatietechnieken worden gebruikt. Het toepassen van simulaties in conceptueel systeem ontwerpen kent echter een aantal uitdagingen. In deze fase is er namelijk nog veel onzekerheid en wordt gewerkt zonder vaste regels en richtlijnen. Daarnaast moet worden gestreefd naar het ondersteunen van multidisciplinaire perspectieven en het ondersteunen van probleem-georiënteerd ontwerpen. Dit onderzoek biedt een raamwerk om met deze uitdagingen om te gaan.

Om de uitvoering en de communicatie van simulaties in de praktijk te onderzoeken zijn er twee case studies uitgevoerd bij de interventionele röntgenafdeling van Philips Healthcare. De eerste case studie onderzocht de hand-oogcoördinatie van een arts, terwijl de tweede case studie het subsysteem energiebeheer analyseerde. De case studies laten zien dat adequate tools en technieken beschikbaar zijn om deze aspecten te modelleren en simuleren. Bijvoorbeeld, door de Y-Chart aanpak in combinatie met de POOSL modelleertaal te gebruiken. Het wordt echter ook duidelijk dat adequate ondersteuning voor de communicatie over de verkregen inzichten ontbreekt. In de eerste case studie is een A3

Architectuur Overzicht gebruikt om de simulatie resultaten te communiceren. Hoewel dit nuttig was, wordt er toch teveel data gegenereerd om in één statisch A3 Overzicht weer te geven. In de tweede case studie is het systeemgedrag gevisualiseerd in een interactief overzicht. In dit overzicht kan het gedrag tegelijkertijd vanuit verschillende perspectieven worden bekeken en besproken.

Gebaseerd op deze case studies presenteert dit proefschrift de COMBOS (COMmuniceer Beter Over Systeemgedrag) methode. COMBOS verankert gedragsanalyses, en vooral multidisciplinaire communicatie van deze analyses, in het ontwerpproces. De methode benadrukt het belang van continue communicatie tussen stakeholders. Het uiteindelijke resultaat van het toepassen van de methode is een interactief systeemoverzicht dat stakeholders in staat stelt om gelijktijdig het systeemgedrag te observeren vanuit verschillende perspectieven. Dit ondersteunt multidisciplinaire ontwerpdiscussies waar bijvoorbeeld ontdekt kan worden dat het gedrag van fysieke componenten veranderd moet worden, gebaseerd op wat er gezien is in het functionele overzicht. Voor het uitvoeren van de methode is een set van elf richtlijnen ontwikkeld.

COMBOS is toegepast in twee voorbeelden. Het eerste voorbeeld is het subsysteem energiebeheer van een elektrische auto. Het tweede voorbeeld heeft betrekking op een key driver, namelijk het tijdsplan van een ruimtevaartmissie. Deze voorbeelden laten zien dat de COMBOS methode toepasbaar is in verschillende situaties en domeinen.

Evaluatie van de methode laat zien dat COMBOS een effectieve aanpak is om te communiceren over systeemgedrag in de conceptuele fase van een ontwerpproces en helpt bij het bepalen van de systeemarchitectuur. Dit wordt bevestigd door een kwalitatieve enquête onder personen die betrokken waren bij de case studies. Deze personen kunnen op dit moment echter nog niet de efficiëntie van de methode beoordelen.

De voornaamste bijdrage van dit onderzoek is het feit dat de COMBOS methode de conceptuele fase van systeem ontwerpen verbetert door te beschrijven hoe analyses van systeemgedrag beter gecommuniceerd kunnen worden. Door deze methode toe te passen zullen conceptuele ontwerpen beter begrepen en gespecificeerd worden vanuit verschillende disciplines. Een verbeterd ontwerp zal leiden tot minder laat ontdekte en dus kostbare fouten. Dit resulteert uiteindelijk in systemen die zowel sneller als goedkoper ontwikkeld zijn en bovenal het gewenste gedrag vertonen voor klanten en eindgebruikers.

Table of Contents

Introduction 1

1

Introduction

This thesis concerns the development of systems and their behavior in particular. This first chapter provides an introduction for the research presented in this thesis. The context for this research is described in section 1.1, by elaborating the role engineered systems play in answering societal challenges. Section 1.2 introduces the importance of understanding the behavior of a system. The research project and research partners are introduced in section 1.3, giving the broader context and goals for this research. Building on this, section 1.4 discusses the motivation for the research presented in thesis. Finally, section 1.5 details the research approach that is employed in this thesis and the chapter is concluded with a description of the thesis structure in section 1.6.

1.1 The Need for Engineered Systems

Current societal issues such as the cost of healthcare pose large challenges for policy makers. [OECD, 2010] states that improving the efficiency of the healthcare system would result in a cost reduction of 2% of the GDP in OECD-countries. However, the report also states that reducing these costs cannot be answered merely by adopting a different type of healthcare system, as the cost-effectiveness of a healthcare system relies on how it is managed.

Within this healthcare system, the management of hospitals focuses on delivering quality care and maintaining a healthy balance between revenues and expenditures. Metrics in this regard are for example the number of complications that occur or the time that patients stay in a hospital. A type of surgery that addresses both of these issues is minimally invasive surgery, also known as laparoscopic or keyhole surgery. It involves making tiny surgical incisions, or "keyholes", to access organs and operate on them. In contrast, traditional surgery involves cutting into and through much larger areas of tissue [Xu *et al.*, 2015]. Xu *et al.* show for the U.S. that if hospitals performing the fewest minimally invasive operations boosted their levels to those of their higher-performing counterparts, they would avert 4.306 complications, reduce hospital stay by 169.819 days and save $337 million a year.

Evidently, an innovation such as minimally invasive surgery can be an important factor in addressing the overall cost of healthcare. However, to execute these types of surgeries, a physician and the supporting staff rely on a number of technologies embedded in medical equipment. For example, in order to "see" both the medical equipment and the ailment of the patient in a patient's body, optics or radiology is required. Optics relies on inserting cameras into a patient's body, while radiology uses radiation such as X-ray to look into the patient's body. During such a procedure, the physician and the supporting staff are continuously required to (i) orient the optics or radiology on the area of interest, (ii) control medical equipment that is inserted into the patient's body, (iii) assess the ailment of the patient and (iv) ensure the safety of both patient and staff. All of these tasks cannot

be supported with a single element. This is why it is required to realize a system that combines different components, technologies and capabilities. To describe such a system, the term "engineered system" is used, as these human-made, technical systems are brought into being to address a certain functional purpose [Blanchard and Fabrycky, 2010].

Many definitions of the concept system exist. These are similar in nature, but have a slightly different focus. The INCOSE Handbook [Walden *et al.*, 2015] defines a system as:

"an integrated set of elements, subsystems, or assemblies that accomplish a defined objective"

[Blanchard and Fabrycky, 2010] state that a system is more than the sum of its component parts, while [Maier and Rechtin, 2000] define a system as "*a set of different elements so connected or related as to perform a unique function not performable by the elements alone".*

The fact that these systems are often termed complex systems stems from the fact that the parts of the system are tightly integrated, or as [Maier and Rechtin, 2000] define complex: "*composed of interconnected or interwoven parts*". The most common approach to the development of complex systems is systems engineering. Systems engineering is defined as "*an interdisciplinary approach and means to enable the realization of successful systems"* [Walden *et al.*, 2015].

Finally, systems do not operate in isolation. They operate in a context in which other technical systems, persons and entities are present. This context is something that should always be considered when developing complex systems. An overarching system that contains a number of these systems is termed a System of Systems (SoS). An operating room in a hospital can be considered as a SoS which contains medical devices, a team of physicians and nurses and basic facilities (lighting, air flow) in the room itself [Kooistra *et al.*, 2014]. All of these are systems of their own. Another example would be the healthcare system, which is an organization of people, institutions and resources that aim to deliver healthcare services.

1.2 System Behavior

As was stated in the previous section, systems are developed to achieve a particular goal. This goal is achieved by specifying functions that a system should be able to fulfil. By establishing elements that are able to execute these functions, a system will be able to fulfil its goal [Maier and Rechtin, 2000]. These functions can be decomposed and assigned in a structure [Pahl *et al.*, 2007]. The combination of a system's structure and behavior enables it to perform its function [Dori, 2011]. Dori defines a system's behavior as its *"varying, time-dependent aspect, its dynamics—the way the system changes over time by transforming objects"*. The concepts of function, behavior and structure are widely recognized and researched. [Gero and Kannengiesser, 2004] define behavior as the attributes that can be derived from the design object's structure, while [Umeda *et al.*, 1990] state that behavior is "*the manner in which a thing acts under specified conditions or circumstances, or in relation to other things"*. These definitions show that behavior is a system aspect that manifests itself during its operation, as opposed to its structure, which can be observed more directly.

A proper understanding of the behavior of a system during its development is vital. In fact, difficulty in predicting system behavior is mentioned as one of the top six challenges in systems design [Boucher and Houlihan, 2008]. The historical trend is that customers and users expect increasingly further capabilities and flexibility of a system. For example, at the end of the 19th century X-ray systems were already used in medical applications [Wikipedia, 2015b]. They were used by physicians to create single pictures to aid in diagnosis. Several decades later, physicians were able to use X-ray systems for interventional procedures, made possible by technological advancements such as the introduction of a C-arm. In recent times, automated control of the C-arm even allows 3D imagery of a subject, instead of the traditional 2D images. This in turn allowed new procedures using 3D guidance to be developed. In the last decades, the flexibility that is exhibited by systems has become more and more realized by the software domain [Basten *et al.*, 2013]. Two examples of this can be found in current interventional X-ray systems such as the Philips Allura system[1]. As a first example, this system includes actuators and sensors that control the motions of the C-Arm and the table on which the patient is positioned. However, new functionality is created in the software domain, by specifying series of motions that together result in a particular visualization useful for a physician. Another example is that the X-ray images that are detected can be processed in a number of ways. While the physical capture of images stays the same, different processing algorithms and techniques result in different images.

Understanding which behavior is desired and which should be avoided is critical to ensure that stakeholder needs are addressed properly. During the design process, this becomes increasingly difficult, because as explained, more and more of the system behavior is hidden in the software domain. [Maier and Rechtin, 2000] state this as the observation that "*architects increasingly need behavioral models as behavior becomes less obvious from the systems form".* This means that it is often not possible to directly observe the system behavior from a description of the system. Use cases diagrams [Object Management Group Inc., 2012], or scenarios [Ionita, 2005] can shed some additional light on the behavior of a system, as these model how a system behaves over time and in varying contexts. These models can for example consider inputs and outputs of the system, to see how elements of the system interact with one another, or with the environment of the system.

This section has shown that the increased flexibility of systems has made system behavior increasingly important, but at the same time less accessible, because it cannot be derived from the systems form directly anymore. Therefore, there is an increasing need to use tools such as models and simulations that are able to capture this behavior.

1.3 The Allegio Project

The research presented in this thesis was carried out as part of the Allegio project [Embedded Systems Innovation by TNO, 2011]. The Allegio project is a collaborative project funded by the Dutch national program COMMIT [COMMIT, 2015]. An overview of the partners is given in Table 1.1.

[1] For an introduction to this system, which also is a subject of case studies, see section 1.3.1)

Table 1.1 – Overview of partners in the Allegio project

Partner	Role
Embedded Systems Innovation by TNO	Lead Partner
Royal Philips	Industrial Partner
Axini	Industrial Partner (Model-Based Testing)
University of Twente	Laboratory of Design Engineering (Group of Author) Laboratory of Design and Analysis of Communication Systems
Technical University Delft	Laboratory of Software Engineering
Technical University Eindhoven	Laboratory of Design and Analysis of Systems

Allegio aims to improve the rate of innovation by researching and applying model-based methods and techniques that support a specification of the design with increased rigor. Model-based techniques should allow designers to uncover extra inconsistencies and errors during the design phase. Part of this approach is early verification of system behavior against the expectations of stakeholders. The end result should contribute to a faster and cheaper development of complex systems that meet the needs of stakeholders better. Applying these methods should allow companies to stay competitive and to address societal problems that can only be solved by designing complex systems that answer these problems.

The Allegio project has four key research themes, of which two are applicable to this research:

- Development of executable models of complex system behavior based on requirements
- Visualization of the execution of models

Furthermore, the Allegio project has also defined its expected results, of which three are:

- Significant reduction in the typical development time of Philips iXR projects
- Exposure of design problems early in the design phase
- Verification of system behavior against the expectations of stakeholders

The research presented in this thesis aims to contribute to these results.

1.3.1 Philips Interventional X-Ray[1]

Royal Philips is the industrial partner of the Allegio project. The involved division is the interventional X-ray (iXR) group of Philips Healthcare. Philips iXR develops so-called angiographic[2] imaging systems that use X-ray radiation to provide images to a physician during interventional cardiac, vascular or neurological procedures. Philips iXR is market leader in this segment.

[1] A note about confidentiality. While the research presented in this thesis reflects the development process at the iXR department of the healthcare division of Royal Philips, abstractions have been made to preserve confidentiality. Statements or conclusions are made from the viewpoint of the researcher and not made by Royal Philips

[2] Angiography or arteriography is a medical imaging technique used to visualize the inside of blood vessels and organs of the body [Wikipedia, 2015a]

Figure 1.1 – Philips Allura Xper FD20 [Royal Philips, 2015a]

One of the key focuses of Philips iXR is to provide images with high clarity, while using a minimal amount of X-ray radiation. Physicians use these images to both diagnose the medical problem of the patient and monitor the position and movements of medical tools inside the patient's body. At Philips iXR, various types of angiography systems are developed. Figure 1.1 shows an example of such an iXR system. The system shown in Figure 1.1 is located in an examination room. However, the complete system also includes a control room in which technicians can monitor and support procedures. Also, the system includes a technical room, in which cabinets containing the power source, computers and other equipment are located. This structure can be seen in Figure 1.2.

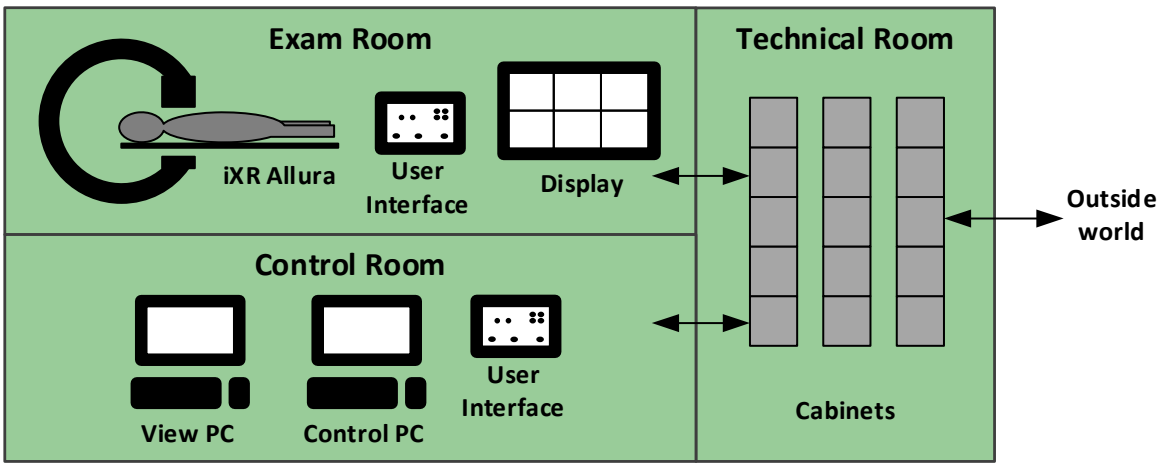


Figure 1.2 – Overview of iXR Allura system distributed across rooms in hospital

1.4 Research Motivation

The previous sections have shown that system behavior is a concept that is difficult to grasp and quantify, but is becoming increasingly important. Models that are able to characterize the behavior of systems are thus important. However, this is only one piece of the puzzle. As tools are used to gain insight in a system's behavior, the next step is to share and discuss these insights across a large group of stakeholders from multiple disciplines. Furthermore, continuous involvement of stakeholders during the modeling and simulation process is important [Morse *et al.*, 2010, Topper and Horner, 2013]. This is why supporting communication of system behavior in an effective manner is crucial.

Models and simulations form an integral part of system design. They allow stakeholders to "*check their own thinking and to communicate their concepts to others*" according to [Walden *et al.*, 2015]. Many types of models and simulations exist. [Walden *et al.*, 2015] classify these as either physical models or abstract models, of which the abstract models can be divided into formal and informal models. These can all be used to capture behavior. Specific to simulations, a report on simulation based engineering science concludes that future engineers must be educated to work with simulations [National Research Council, 2002]. They must also be allowed to develop communication skills necessary for effective development and deployment of simulation based engineering and science in multidisciplinary research teams. While the report recognizes communication as a skill that must be cultivated, it does not explicitly recommend research into supporting this process.

Sometimes, it is difficult to conceptualize ideas, especially when dealing with complex systems. Models help in the process of making these tacit ideas explicit [Nonaka and Takeuchi, 1995], and allow their communication. In conceptual system design, these models often take the form of drawings and simple diagrams [Bonnema, 2008]. These drawings and diagrams are often created on a personal basis, to clarify concepts for an internal comprehension process. However, they might be even more important in a communication setting, where they are used to share ideas and clarify concepts. They for example serve as a common reference blueprint during design [Topper and Horner, 2013]. Most drawings and diagrams mainly give information on the structure of a system, whereas its behavior is often not yet characterized in these early stages. In later stages of design, further formal models arise, for example CAD/CAM models (mechanical engineering) or UML diagrams (software engineering). These models allow simulation and can give an indication of a system's behavior. However, these models are less likely to be used in multi-disciplinary design discussions because these models stem from a mono-disciplinary background.

Currently, several multi-disciplinary modeling approaches exist [Canedo and Richter, 2014, Dori, 2002]. While these models are easily understandable and relate to the conceptual system design stage better because of abstract or functional reasoning, they have been developed as a as a modeling approach, but not as a communication tool. An approach that aims to support communication is the A3 Architecture Overview (A3AO) method [Borches, 2010]. However, the A3AO method focuses mainly on communicating the systems structure, and does not allow a user to explore the system's behavior.

The motivation for the research presented in this thesis is based on the fact that characterizing and understanding system behavior using models and simulations is widely recognized as important. Moreover, it will become increasingly important in the future. Models and simulations are said to support communication by making concepts explicit. Nonetheless, it is still difficult to communicate these concepts, especially across discipline boundaries. This might be because this part of communication is viewed as a skill that is mastered through practice, instead of something that can be supported. In conclusion, the research in this thesis will explore supporting multidisciplinary discussions on the behavior of a system during its design. This is especially important in the conceptual system design stage, as the multidisciplinary aspect is most prevalent there.

1.5 Research Approach

This section discusses the approach that is used for the research in this thesis. The research approach is structured following the Design Research Methodology (DRM) from [Blessing and Chakrabarti, 2009] and can be seen in Figure 1.3. Research methods are also structured following the research presented in [Braa and Vidgen, 1999] (see also Table 1.2). Furthermore, as this research is positioned within the Allegio project and the main industrial partner is the interventional X-ray division of Royals Philips Healthcare N.V., the case studies will be focused on this partner as well. Additionally, the Allegio project supports the so-called Industry-as-Laboratory approach [Muller, 2010]. Borches [2010, p6] identifies that this differs from traditional case studies due to the close collaboration between researcher and practitioners, where the practitioner might even guide, mentor and support the researcher. In this research, this has partly been the case during the first two phases (research clarification and descriptive study I). [Potts, 1993] states that the Industry-as-Laboratory approach is crucial in ensuring that the problem definition can be based on empirical research and that the interaction between technical and nontechnical factors becomes part of the subject matter of the research. Additionally, the research in this field is often trademarked by the fact that the researcher performs multiple roles. First off all, the researcher is a researcher, who defines and answers research questions and observes the process to extract experiences, statements and knowledge. On the other hand, the researcher is also a practitioner, who collects data, designs tools and solves real-world problems [Muller, 2009]. This can have both a negative and positive effect on the research, which is described well by [Borches, 2010, p7, table 1.1].

The overall research approach consists of the following four phases that were identified and structured using DRM.

The **Research Clarification** stage focuses on creating an initial description of the situation through a literature review of the general areas of interest that have been identified so far. The relevant research areas that will be explored are conceptual system design and three specific areas within conceptual system design: modeling & simulation, communication and decision making. Decision making has not been explicitly mentioned so far in the introduction, but it is important as an integral part of communication in conceptual system design deals with decision making. These areas will be considered both

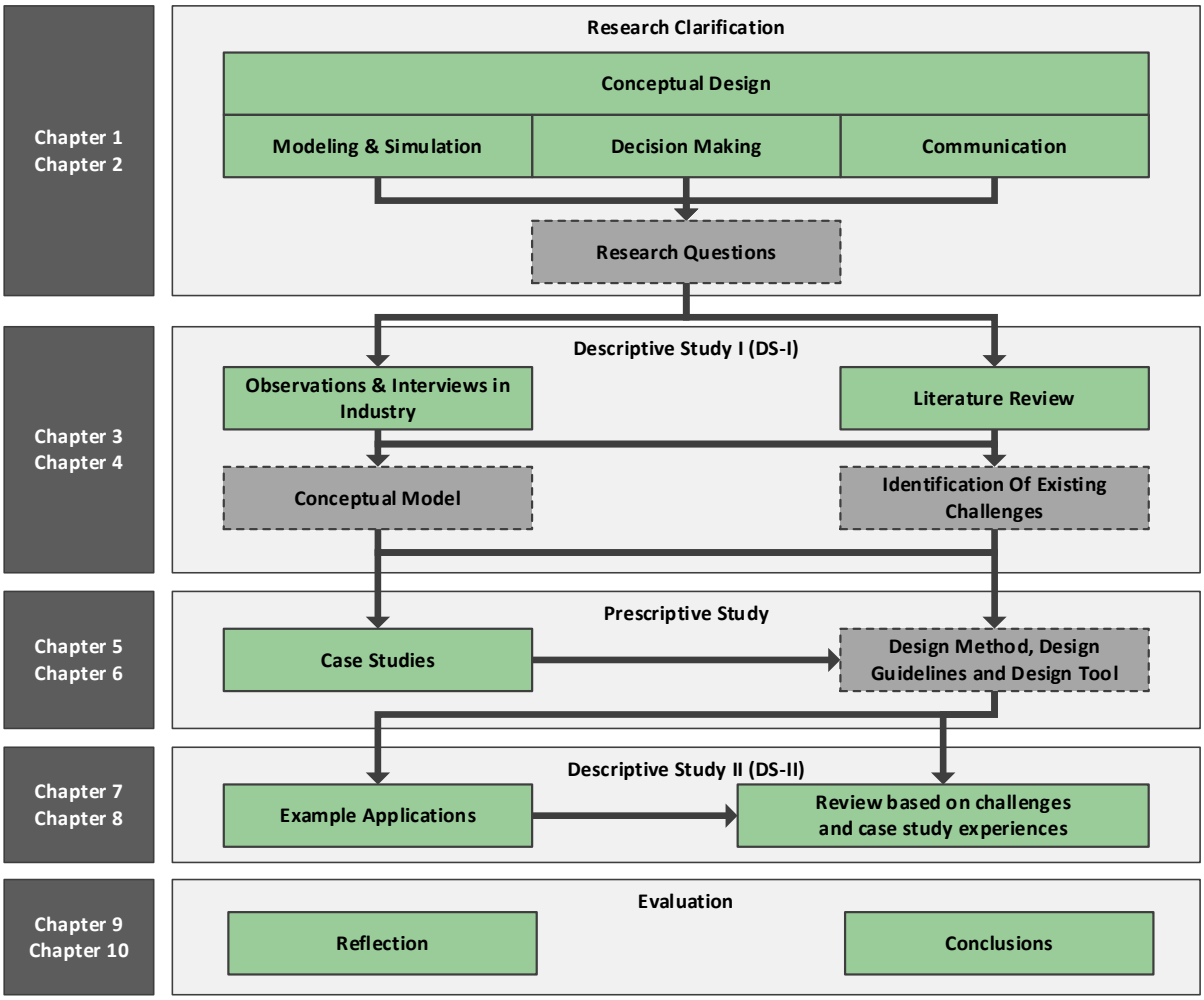


Figure 1.3 – Research approach and structure for the research in this thesis. Research activities have solid lines, while outputs are darker and have dotted lines.

from a theoretical foundation, as well as reported experiences from practice in literature. This stage results in a definition of research questions that frame the further research.

The next stage is the **Descriptive Study I** stage, which aims to elaborate the existing situation. This will be done by both further literature review aiming to answer research questions, as well as by observations and interviews in industry. The observations in industry will take the form of soft case studies [Braa and Vidgen, 1999], see also Table 1.2. This includes the researcher taking part in day to day activities at Philips Healthcare. The interviews will initially focus on Philips Healthcare. However, other industries will be approached to validate identified issues at Philips Healthcare. The interviews are structured as free-flow interviews, with a focus on qualitative data collection instead of quantitative data collection. [Muller, 2013] states that such a free format works well in discovery and exploratory phases. This stage should result in a collection of important concepts in a reference model, as well as identification of challenges that need to be addressed to be able to support communication on system behavior.

The third stage concerns a **Prescriptive Study**. In this stage, case studies are utilized to define the intended support and adapt it in such a manner that this results in a final, actual

support. The case studies that are used in this stage are action cases (see Table 1.2), as there is medium participation of organizational stakeholders. Moreover, the case studies have as goal to improve the design of the system at Philips Healthcare. Not only by the immediate results of the case study, but also through the development of a support.

The fourth and final stage is the **Descriptive Study II**. In this research this consists of example applications of the developed support method. These examples can be characterized as hard case studies (see Table 1.2) as they mainly serve as activities that aim to increase understanding of the application of the method. They also serve as prediction to whether the method can be applied. The examples focus on usability and applicability of the method by extending the scope of the method to other domains and discussing this. The research presented in this thesis does not include case studies that aim for success evaluation.

In terms of the type of research that will be done according the to the DRM framework, it consists of a review-based research clarification, a comprehensive descriptive study I and prescriptive study, and finally an initial descriptive study II. This corresponds with the fifth type of design research as described in [Blessing and Chakrabarti, 2009, p61].

In the industry-as-laboratory approach, validation often stems from the fact that outcomes are actually applied in industry [Muller, 2010]. Furthermore, validation is based on a review based on identified challenges in the DS-I phase and a survey to obtain feedback from involved stakeholders. By using multiple validation sources from different perspectives, the confidence in the results of the overall validation will be increased. Furthermore, a number of common pitfalls in validation are listed by [Muller, 2013]. For example, one of those is that an enthusiastic survey response does not mean directly that a method is "good".

Table 1.2 – Research method characteristics, obtained from [Braa and Vidgen, 1999, p61]. The research methods field experiment and quasi field experiment are omitted.

		Research method			
		Hard case study	**Soft case study**	**Action research**	**Action case**
Research Outcome	**Change (intervention)**	*Unintended*	*Unintended*	*Intended, large scale*	*Intended, small to medium scale*
	Prediction (reduction)	*Medium*	*Low*	*Low*	*Low*
	Understanding (interpretation)	*Medium*	*High*	*Low to Medium*	*Medium*
Research Characte-ristics	**Duration**	*Any*	*Any*	*Long*	*Short to Medium*
	Time orientation	*Contemporary*	*Historic and contemporary*	*Building Future*	*Contemporary and building future*
	Participation (organization)	*Low*	*Low*	*High*	*Medium*

1.6 Thesis Outline

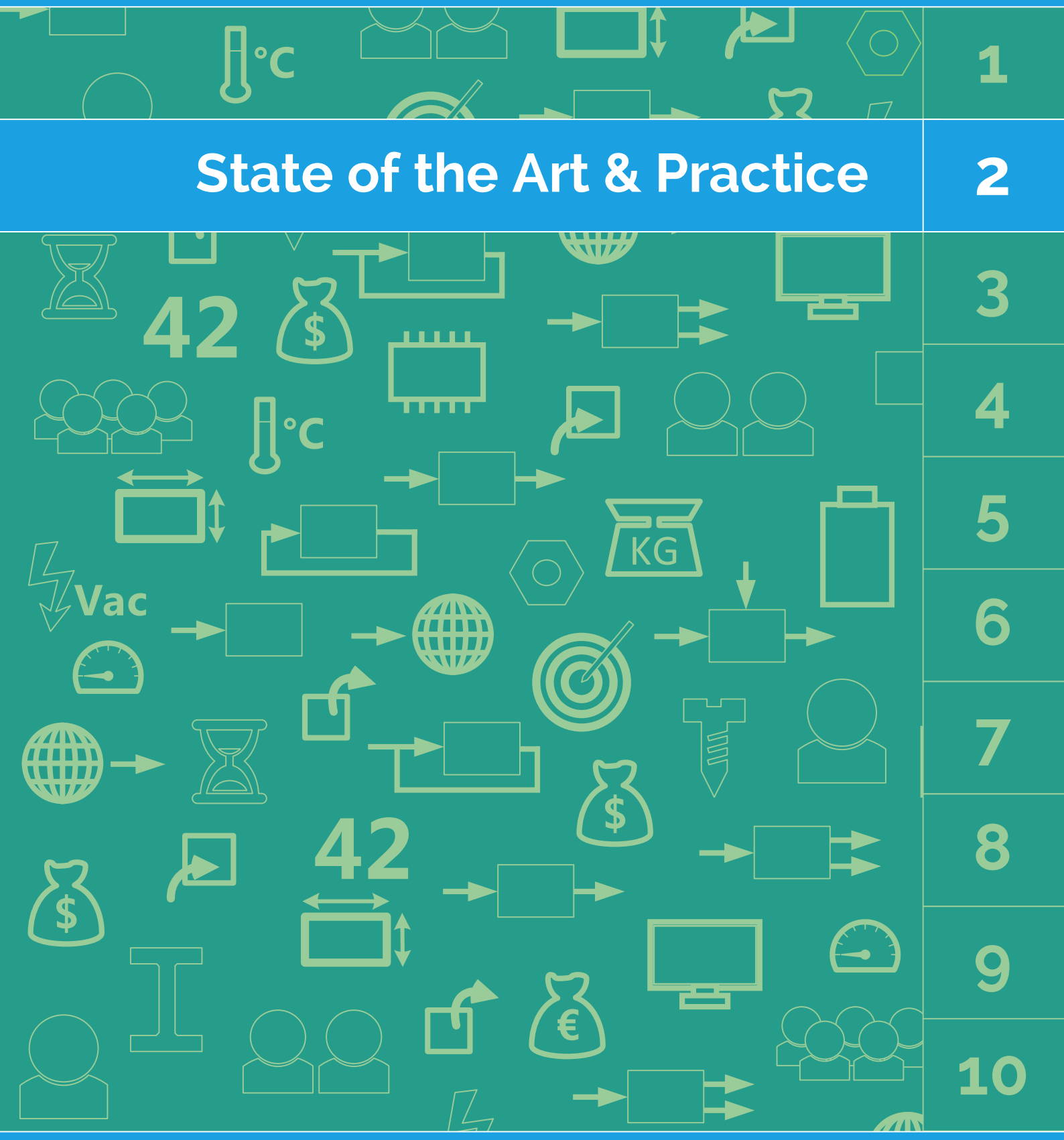
The thesis is structured according to the research approach, presented in Figure 1.3. Note that the evaluation in Chapter 9 and Chapter 10 considers the complete research approach. Below, the chapter outline is further clarified.

The research motivation given in Chapter 1 introduces a basis for the exploration of the state of the art & practice in Chapter 2. Chapter 2 explores conceptual system design and the topics communication, decision making and models & simulations within this context.

The first contributions of this thesis are presented in Chapter 3 and Chapter 4 where an in depth literature review is combined with interviews and observations in industry. Chapter 3 explores system behavior in conceptual system design and identifies the views that are required to represent system behavior in conceptual system design. Chapter 4 presents key challenges that need to be addressed to enable effective communication. It also presents a framework for simulations in conceptual design as well as a reference model that summarizes and ties together the state of the art and practice.

Based on the identified views and challenges, the communication of system behavior in practice is addressed in Chapter 5. Two case studies at the interventional X-ray division of Royal Philips are used to explore this issue. Based on this exploration, Chapter 6 presents the COMBOS (COMmunicating Behavior Of Systems) method. COMBOS aims to consistently embed behavioral analysis in the conceptual system design phase. The COMBOS method is applied in Chapter 7 by using it in two example applications. Chapter 8 evaluates the COMBOS method, based on the challenges identified in Chapter 4 coupled with a survey on stakeholder experiences with the method.

Finally, Chapter 9 reflects on the research presented in this thesis and reviews the contributions of this thesis, as well as the research approach and validity of results. The conclusions as well as recommendations for further research are given in Chapter 10.



State of the Art & Practice

2

State of the Art & Practice

This chapter reviews the state of the art and practice in literature and identifies issues that are relevant for this research and to support the development of research questions. The state of the art concerns the theoretical foundations of the various concepts that will be treated in this chapter. It also considers how these theories can be applied through methods and tooling. The state of the practice concerns the perspective of reported experiences with applying these methods and tools in industry. It also discusses studies that try to directly identify issues existing in industry.

This chapter considers five topics. Section 2.1 discusses conceptual system design from an overall perspective. Systems architecting is treated in more detail in section 2.2. Communication (section 2.3), decision making (section 2.4) and models and simulations (section 2.5) are all considered in the scope of conceptual system design. The chapter is concluded by distilling the literature review into research questions in section 2.6.

2.1 Conceptual System Design

The early stages of system design are also termed the conceptual phase. The INCOSE Handbook [Walden *et al.*, 2015] separates the conceptual phase in a conceptual stage and an exploratory research stage, whereas [Blanchard and Fabrycky, 2010] consider this one phase. Conceptual system design is an activity taking place in that phase. This section aims to give a short overview of the foundations of conceptual systems design and the concepts and philosophy behind them.

2.1.1 Design Theories and Methodologies

Multiple theoretical foundations are relevant to the conceptual system design process. A recent and comprehensive review by [Tomiyama *et al.*, 2009] distinguishes between design theories and design methodologies, but considers the field as one entity.

Axiomatic Design [Suh, 1990] defines complexity in the functional domain based on mathematical relationships. It strives for "simplicity" by relating functional requirements to design parameters and giving two axioms for their relation. The first axiom, the independence axiom, states that functional requirements should be independent. Ideally, a design parameter is thus only linked to a single functional requirement. The second axiom, the information axiom, states that the information content of the design should be minimized. Suh extended this theory with an application to engineering [Suh, 2005]. An example of an additional axiom is *"minimize the number of functional requirements"*. Suh also distinguishes time-dependent and time-independent complexity.

The General Design Theory [Tomiyama and Yoshikawa, 1986] considers design from the perspectives of the real world, the conceptual world and the logical world. These are based

on three design situations that have been identified by [Alexander, 1964], and are also discussed in [Bonnema, 2008, p13]. The theory itself is based on three axioms which consider recognition, correspondence and operation. Ultimately, applying the theory to for example computer aided design shows that these tools should embed mechanisms to represent physics-centered modeling, function modeling and finally also intention modeling [Tomiyama, 1994].

Most of the methodologies described in [Tomiyama *et al.*, 2009] describe a process that can be applied generically. This includes design methodologies such as TRIZ [Altshuller, 1997] and Pahl and Beitz [Pahl *et al.*, 2007], methodologies that aim to achieve concrete goals as for example Quality Function Deployment [Mizuno and Akao, 1994] and process methodologies such as Concurrent Engineering [Sohlenius, 1992]. [Bonnema, 2008] mentions several of these design models and discusses their meaning for the conceptual system design process.

[Pahl *et al.*, 2007] describe a systematic approach, which specifies the conceptual system design process in several steps. They start with an initial problem abstraction that allows a definition of the basic functions. These functions are decomposed and working principles for the functions are defined. These ultimately result in various solutions that are evaluated, resulting in a principle solution (concept). In this methodology there is an implicit distinction between problem and solution domain, and abstraction is an important consideration. Altshuller's TRIZ [Altshuller, 1997] also considers abstraction. A specific problem is abstracted into a generic problem for which a generic solution can be found. This generic solution can be concretized into a specific solution for the case at hand.

Concurrent Engineering can use the principles of set based design [Sobek *et al.*, 1999]. Instead of a point-based solution definition approach as in many other design methodologies, stakeholders in the design process work with a solution space. This allows users to work concurrently. Fast feedback cycles are important to support this methodology. Sharing of design intent, constraints and other life cycle information is critical.

[French, 1985], which is briefly mentioned by [Tomiyama *et al.*, 2009], but treated in more detail in [Bonnema, 2008] specifies the conceptual design process with four concrete results and intermediate steps. The start is a need, which via an analysis of the problem results in a statement of problem. Conceptual system design transforms this problem statement in several selected schemes. The embodiment and elaboration of these schemes results in a set of drawings or other detailed design deliverables.

A last process model that is worth mentioning is SIMILAR [Bahill and Gissing, 1998]. SIMILAR is an acronym for: State the problem, Investigate alternatives, Model the system, Integrate, Launch the system, Assess performance, and Re-evaluate. SIMILAR explicitly covers both spiral and staged development models [Woestenenk, 2014]. Furthermore, an explicit step in the process is to model the system (the M in SIMILAR), which is interesting as this is not explicitly described in most other design methodologies.

As a final note, the systems engineering process as described by [Blanchard and Fabrycky, 2010] can be used to perform conceptual system design as well. They provide a prescriptive, process-oriented approach. They coin the term system design considerations as *"the full range of attributes and characteristics that could be exhibited by an engineering*

system, product or structure. These interest both the producer and the customer". They ultimately present a design morphology in which they discuss the fact that design synthesis is both a top-down approach as well as a bottom-up approach. This is supported by the fact that they recognize that both a customer need, but also available technologies and system data are inputs for this process.

2.1.2 Development Models

To structure the systems engineering process, and with it the conceptual system design process, various development models have been established. These are the waterfall model [Royce, 1970], the spiral model [Boehm, 1986] and the Vee-model [Forsberg and Mooz, 1991] (see also Figure 2.1).

When looking at systems engineering development processes, it can be observed that many projects overrun both schedule and cost [Maier and Rechtin, 2000, Walden *et al.*, 2015]. This is often caused by errors in the design that are only found later in the testing and integration process. Referring to Figure 2.1, errors made in the left side of the Vee-model are only discovered in the right side of the Vee-model. If errors are detected during testing, they are costlier to mitigate [U.S. Department of Transportation, 2007]. The early discovery of errors through a more in-depth analysis of the system's behavior during the conceptual system design stage is exactly what this research tries to address. Therefore, the Vee-model is used as a reference, because it allows a good representation of this issue. Important to note is that while the number of issues addressed in the top of the Vee-model is limited [Muller, 2011a], they have to be considered across multiple disciplines. In detailed design, the number of issues that are considered increases considerably, but it can also be afforded to stay mostly within the boundaries of a discipline.

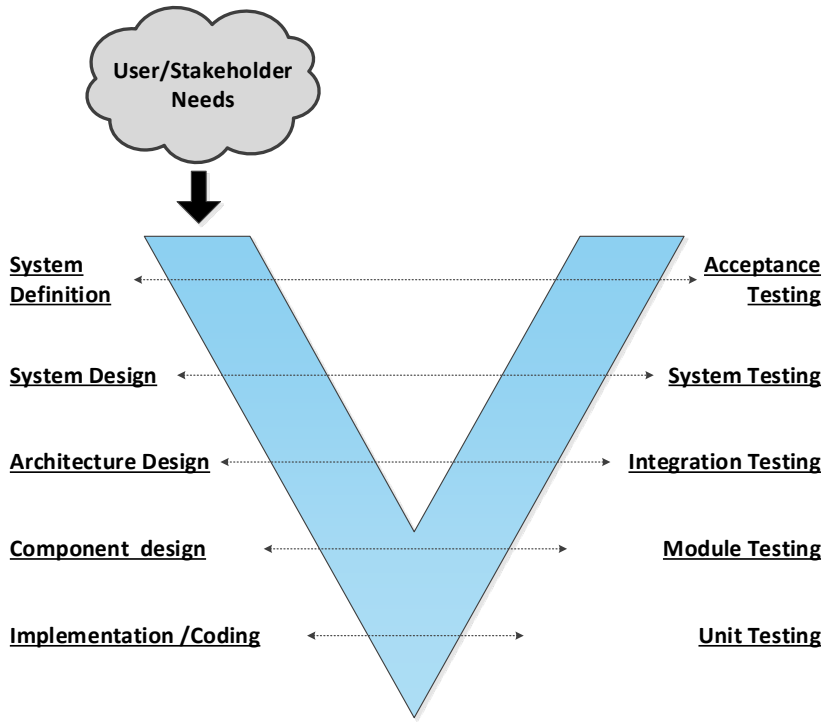


Figure 2.1 – Vee-model based on [Forsberg and Mooz, 1991]

2.1.3 Conceptual System Design in Practice

Conceptual design of complex systems is a multidisciplinary activity by nature. [Tomiyama *et al.*, 2007] note that three types of issues arise when experts in different disciplines collaborate: *"(i) there is no common inter-disciplinary design language (an ontology problem), (ii) there are inherent difficulties in dealing with many stakeholders during the design process; and (iii) multi-disciplinary product development creates inter-disciplinary problems"*. These issues relate to the respective fields of communication, decision making (which are both discussed later in this chapter) and finally design in general.

Considering the conceptual system design, [Bonnema *et al.*, 2015] discuss a practical implementation. The design is framed as a constant loop of investigating the problem and defining the solution. [Bonnema *et al.*, 2015] also state:

"Many engineers think in solutions. They even investigate the problem using a solution ... when the distinction between problem domain and solution domain is not made, jumping to conclusions, and picking the first solution that comes to mind become real dangers"

TRIZ [Altshuller, 1997] is a methodology that explicitly considers the problem domain. TRIZ is considered to be a method that can lead to creative and elegant solutions [Bonnema, 2008]. An interesting observation is that there seems to be a correlation between reasoning in the problem domain and the ability to find creative solutions that reduce complexity.

Another interesting discussion point is how the application of design methods in practice results in collaboration between stakeholders. For example, it is well known that the traditional system design process as described by [Blanchard and Fabrycky, 2010] or [Pahl *et al.*, 2007] results in a plethora of deliverables and documentation. This results in the need for tooling that manages these design artefacts and ultimately also becomes a basis for communication. Furthermore, relevant architectural knowledge is obfuscated in this massive amount of documentation [Muller, 2004].

Concurrent Engineering aims to circumvent these issues by colocating stakeholders, especially in the conceptual system design stage. Collaboration environments such as the Concurrent Design Facility (CDF) [ESA, 2015], JPL Team X [Mark, 2002], Collaborative Design Environment (CODE) [Osburg and Mavris, 2005], Skunk Works [Lockheed Martin, 2015] overlay *"a social structure on technical models to improve communication and reduce feedback delays"* [Kolfschoten *et al.*, 2012]. In these environments, a movement from traditional "Excel-based" towards Model Driven design can be observed [Schumann *et al.*, 2010].

The Boderc design process [Heemels *et al.*, 2006b] focuses on the most critical issues of a system and aims to use simple models that create insight in a design decision. It considers a design preparation phase, a selection phase where critical design aspects are identified and an evaluation phase in which small models are built and measurements are performed.

2.1.4 Conclusions

From the review of conceptual system design theories and their application in practice, several conclusions can be deduced. First, it is clear that considering the problem domain and

sufficiently exploring it contributes to a better system design in the end. This also reduces the number of issues that are found in the testing and integration phase. To represent this opposition, this research uses the Vee-model as a reference. Second, abstracting the real world with conceptual representations is an important step in conceptual system design. Finally, supporting collaborations between stakeholders is crucial and therefore this is one of the aspects that will be discussed in more detail in the following sections.

2.2 Systems Architecting

During the conceptual system design stage, a system architecture is defined. However systems architecting is not synonymous with conceptual system design, as it can be employed over the complete systems lifecycle and thus during the complete systems engineering process. Systems engineering was already defined in section 1.1. In the well-known book on systems architecting, the Art of Systems Architecting, [Maier and Rechtin, 2000] compare engineering and architecting as follows:

"Engineering is a deductive process that deals with measurables stemming from analytical tools... Architecting deals with unmeasurables, using non quantitative tools ... and is an inductive process.... Engineering aims for technical optimization, architecting for client satisfaction".

They also remark that systems engineering is a science, while systems architecting can be considered an art, hence the title of the book. [Blanchard and Fabrycky, 2010] mention system architectures very briefly, and do not coin the term systems architecting. They do state that the system architecture deals with both requirements and structure and is enabled by a functional analysis of the system and a functional allocation [Blanchard and Fabrycky, 2010, p78]. In [ISO/IEC/IEEE, 2011], a system architecture is defined as:

"The fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution"

Furthermore, systems architecting is defined in [ISO/IEC/IEEE, 2011] as:

"A process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle".

A definition that really defines what systems architecting constitutes is given by Bonnema in [Bonnema, 2008], based on [Gulati and Eppinger, 1996]. In this work, a system architecture is defined as:

"A system architecture defines the parts constituting a system and allocates the system's functions and performance over its parts, its user, its super system and the environment in order to meet system requirements."

The formal and methodology independent definitions for systems architecting and an system architecture given in [ISO/IEC/IEEE, 2011] are useful when discussing ontologies and relations between frameworks. However the definition given by Bonnema is more useful

for practical applications, as it aligns with existing conceptual system design methodologies. It can also be concluded that systems architecting is one of the key activities when defining a system in its early stages. The early stages are also the stages where the issues present require the involvement of stakeholders from multiple disciplines, opposed to the often more mono-disciplinary detailed design stages. Therefore, communication and knowledge sharing are important issues in systems architecting. This has also been observed in previous research [Haveman, 2009], where system architect needs such as *"deliver the right information to stakeholders while keeping the irrelevant part of the information low"* and *"make sure that information is conveyed and interpreted correct"* were identified (see also Table 2.4).

2.2.1 Supporting Systems Architecting

The next question is how the creation of system architectures is currently supported. [Haveman, 2009] lists a number of supportive tools such as requirements management tools or project lifecycle management tools and states:

"these tools focus on phases in the project where the architecture and/or requirements are more or less defined. The early, conceptual stages of a project are not supported well by these tools".

[Bonnema, 2008, p6] states that the current state of support of systems architecting is minimal. This is also indicated by [Muller, 2004]. Bonnema himself proposes FunKey architecting, a method to support the definition and expression of a system architecture by coupling key drivers to functions. [Muller, 2004] on the other hand presents CAFCR, a multi-view framework for systems architecting throughout a system's development. [Muller, 2004, p45] also lists a number of patterns that are applied by system architects such as viewpoint hopping and using a problem solving approach.

Furthermore, [Muller, 2004] presents a method to link user needs to requirements by using key drivers. Key drivers capture the essence of the objectives of the customers, and can be regarded as top level generalized requirements. Mapping requirements to key drivers [Heemels *et al.*, 2006a] allows clear communication and negotiation. Muller also mentions that the key-driver method shows similarities with goal-oriented requirements engineering [Lapouchnian, 2005]. Finally, [Muller, 2004] mentions in the evaluation of the CAFCR method that some engineers experienced that a gap remains between system level specifications (high abstraction level) and module level implementation (low abstraction level). [Muller, 2004] refers to this as the balance between genericity and specificity and suggests that fast iterations using "threads of reasoning" between the two are required to preserve this balance.

2.2.2 Systems Architecting Frameworks

While specific tools and methods might be lacking, frameworks describing the system architecture for a system are numerous. As the notion of a system and its architecture is not limited to engineering, various frameworks with different applications exist. These frameworks describe how to construct architecture descriptions and what constitutes them. A

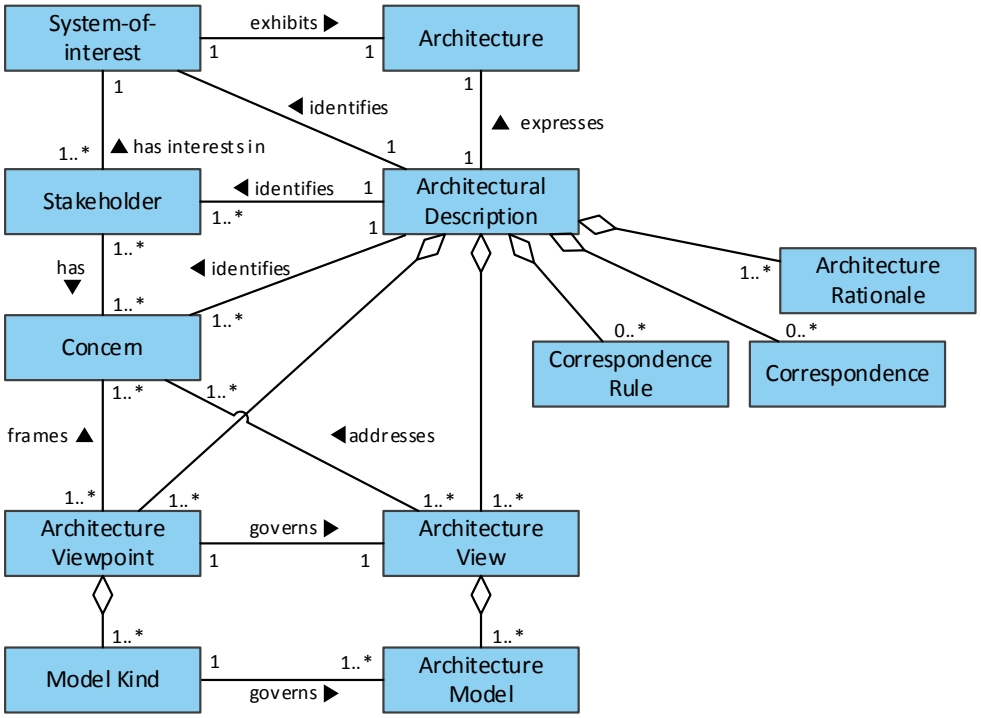


Figure 2.2 – ISO/IEC/IEEE 42010 conceptual model of an architecture description reproduced from [ISO/IEC/IEEE, 2011]

good review of architecture frameworks is given in [Reichwein and Paredis, 2011]. A few of the architecture frameworks they describe will be re-iterated here.

The ISO/IEC/IEEE 42010 standard [ISO/IEC/IEEE, 2011], an update from IEEE 1471 [IEEE, 2007], is a software intensive system oriented architecture framework. It mainly provides an ontology for the concepts that compose the architecture framework. It does so by connecting these concepts in a conceptual model of an architecture description, which is reproduced in Figure 2.2. However, it lacks a specification of which views should be used.

The Department of Defense Architecture Framework (DoDAF) version 2.0 [U.S. Department of Defense, 2010] defines a number of viewpoints that constitute an architecture description. These viewpoints used to be termed views in DoDAF version 1.5 but were aligned to the concept of an architecture viewpoint as defined in [ISO/IEC/IEEE, 2011] in DoDAF version 2.0. The DoDAF framework consists of eight viewpoints. For each of these viewpoints, a number of models is described. For example, the operational viewpoint has specified 9 models of which one is OV-5b, the operational activity model.

The 4+1 method of Kruchten is aimed at software architectures. As the name indicates, five views are used to describe the system. The four basic views are the logical view, the development view, the process view and the physical view. Kruchten argues that a fifth view is required to illustrate the system, which he calls scenarios. The fact that this fifth view has to be represented across the other four views is an important concept that will be re-used in this thesis. However, it must be noted that views in the framework of Kruchten are viewpoints in [ISO/IEC/IEEE, 2011].

Table 2.1 – Zachman Framework with views (row) and viewpoints (columns) [Zachman, 2008]

	What	How	Where	Who	When	Why
Executive Perspective (Contextual)						
Business Perspective (Conceptual)						
Architect Perspective (Logical)						
Engineer Perspective (Physical)						
Technician Perspective (Detailed)						

Finally, the Zachman Framework [Zachman, 2008] is a framework originally developed for enterprise architectures. It constitutes a two dimensional matrix with a view on each row and a viewpoint on each column, see also . These viewpoints consider the views with primitive interrogatives: What, How, When, Who, Where and Why. These viewpoints can be considered as concerns in [ISO/IEC/IEEE, 2011]. [Muller, 2004] applies these questions directly as an architecting method. The views in the Zachman Framework should be considered as viewpoints in [ISO/IEC/IEEE, 2011].

Throughout the text, the concepts in DoDAF, the 4+1 method of Kruchten and Zachman were all compared with the ISO/IEC/IEEE 42010 framework to align terminology. This is done because the architecture definition given in the ISO/IEC/IEEE 42010 standard and its associated definitions will be used throughout this research. First of all, this is done to ensure consistency in the presented research. Second of all, ISO/IEC/IEEE 42010 was chosen because it is domain independent (instead of software or enterprise oriented), resulting in a clear ontology that can be reused in this project. Another important reason is that ISO/IEC/IEEE 42010 explicitly defines both stakeholders and concerns, something that is lacking in other views such as DoDAF [Emery and Hilliard, 2009, Reichwein and Paredis, 2011].

2.3 Communication in Conceptual System Design

In order to share knowledge and discuss a design, communication is required. [Boardman and Sauser, 2008] even state:

"Communicating strategic intent cannot be entrusted to writings alone, nor to the presentations of the author. Additional support is needed—support that is faithful to the statements expressing the strategic intent, but value adding".

This quote emphasizes that communication cannot be taken for granted. During the conceptual system design stages multidisciplinary communication plays a key role [Bonnema, 2014] and is especially relevant during the conceptual stage. To show the need for the support that [Boardman and Sauser, 2008] mention, this section will first take one step back and consider the communication process from a theoretical perspective and review how this theory is reflected in practice.

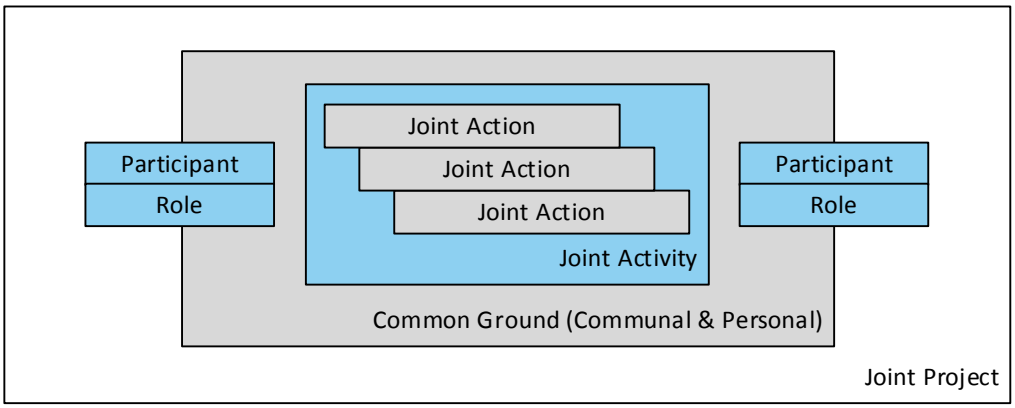


Figure 2.3 – Overview of concepts describing communication activities in [Clark, 1996]

2.3.1 Theory

Various theoretical foundations for communication exist. In this case, the work of [Clark, 1996] is used as a start to explore these foundations. [Clark, 1996] reasons from the use of language and introduces three foundations, which are joint activities, joint actions and common ground.

These foundations will be discussed from a conceptual system design perspective and are visualized in Figure 2.3. Imagine a design discussion between two persons on a medical system. This is what Clark calls a *joint project* which is based on a *joint activity* and this is the setting for communication. Clark states that joint activities have *participants* that fulfil a certain *role*. For example, one participant could be a system architect, the other the end-user of the system. All participants also have a *goal* for a particular activity. For this particular meeting, the goal of the system architect might be to uncover the needs of the end-user, while the end-user might have no particular goal, but simply wants to accommodate the system architect. Together, the participants will seek to achieve their goals in incremental steps through *joint actions*. These joint actions require *coordination* on both content and process. However, to achieve this coordination, participants require the assumption of *common ground*. Common ground is the frame of reference that participants have established together. Clark describes two types of common ground, communal and personal. Communal common ground is based on a cultural frame of reference, while personal common ground is based on knowledge from previous interactions. In this example, the end-user might be a physician. Therefore, the system architect can assume that medical terms can be used in the conversation. This would be an example of communal common ground. The physician might have experienced in a previous design discussion with the system architect that the architect likes the use of explanatory drawings. Therefore, the physician might use more drawings based on the personal common ground. Common ground remains the central topic in the rest of Clarks work. He furthermore states that *"in conversations, speakers don't just speak and listeners listen"*. This implies that in communication, a constant feedback cycle occurs. [Heer and Agrawala, 2007] add to this that common ground should be supported in collaborative visualizations.

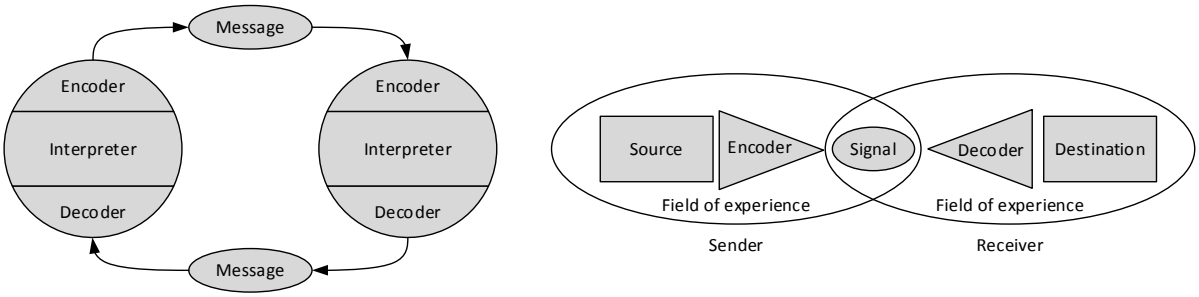


Figure 2.4 – Schramm's model of communication [Schramm, 1954]. The left side shows the feedback model and the right side the field of experience concept.

Another communication model that focuses on the process of message sending is Schramm's model of communication (see Figure 2.4) [Schramm, 1954]. Here, the aforementioned feedback cycle is visualized in the left side of Figure 2.4. The right side shows the concept of field of experience. The overlap between the fields of experience is what Clark refers to as common ground. It shows the participants as processors of messages that are decoded interpreted and finally a new message is encoded and sent to one or more recipients, with whom the same process repeats itself. Furthermore, the Shannon-Weaver model of communication [Shannon and Weaver, 1949] must be mentioned as well, as it introduces the concept of noise that a signal (in Schramm's model) can experience during transmission.

While the communication process has been discussed, the issue remaining is the purpose of the communication. In general, communication in conceptual system design will focus on sharing and creating knowledge. The theory of organizational knowledge creation [Nonaka and Takeuchi, 1995] offers an excellent framework to categorize these knowledge creation processes. An overview of these processes can be viewed in Table 2.2.

Socialization is about sharing experiences, doing things together and learning by observation. The classical apprenticeship comes to mind, but also the colocation that is propagated in concurrent engineering facilities such as the ESA CDF [ESA, 2015]. *Externalization* makes knowledge explicit. Designers can do this through models, be it textual, graphical or mathematical variants. *Internalization* is represented by for example reading documentation, and gaining knowledge from that. Internalization is generally not considered to be a sufficient means to understand a system, as the quote from [Boardman and Sauser, 2008] in the introduction of this section showed. Another example of the perceived value of documentation to understand a system is given in Figure 2.5. Finally, *combination* is the creation of new knowledge by combining various explicit data sources. For example reviewing a design specification against a requirements specification leads to the creation of an evaluation report.

Table 2.2 – Overview of knowledge creation processes [Nonaka and Takeuchi, 1995]

	To Tacit Knowledge	**To Explicit Knowledge**
From Tacit Knowledge	Socialization	Externalization
From Explicit Knowledge	Internalization	Combination

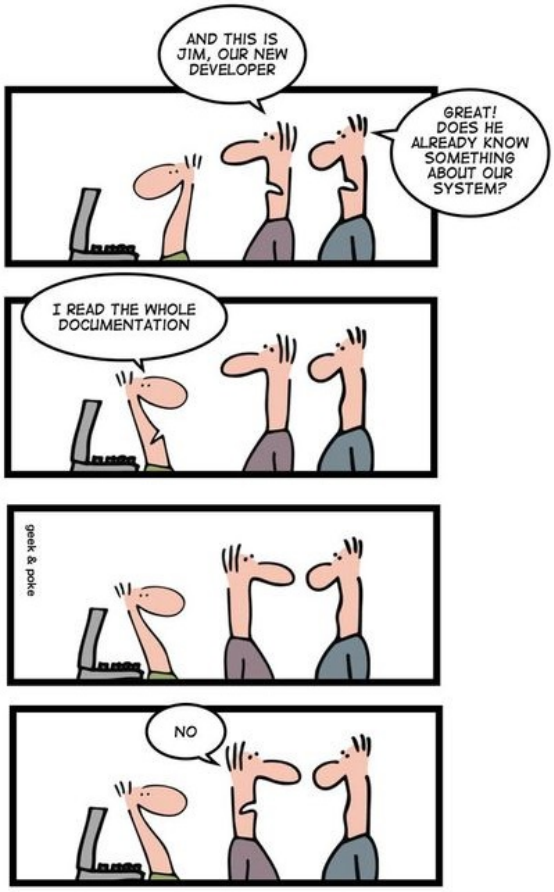


Figure 2.5 – Illustration of the perceived value of documentation to gain knowledge about systems (obtained from http://geek-and-poke.com/ with permission from the author)

In conclusion, a person transmitting a message will compose this message based on the current repository of design knowledge that is both explicitly and tacitly available to the sender of information. The sender encodes this message in the form of speech, a model or written text. The receiver decodes this message, based on the design knowledge that is both explicitly and tacitly available to the receiver. Each time this cycle occurs, new information becomes available either due to the interaction in a discussion or by generation of new knowledge during activities outside of these discussions. This expands the common ground between participants, allowing additional interactions that were not possible at first.

2.3.2 Practice

When the current challenges in systems engineering are considered, communication is a central theme. For example, Torry-Smith et al. [Torry-Smith *et al.*, 2011] present a comprehensive literature review in which they identify nineteen reported engineering challenges. Of those nineteen, at least ten deal directly with communication. These can be seen in . Furthermore, companies that are classified as best-in-class in system design undertake several actions to improve their design. However, the top action is to improve communication and collaboration across disciplines [Boucher and Houlihan, 2008].

Table 2.3 – Multi-disciplinary communication challenges, based on (Torry-Smith et al., 2011)

1	Lack of a common understanding of the overall system design
2	Lack of a common language to represent a concept
3	Transfer of models and information between domains
4	Different traditions within domains for how to conduct creative sessions
5	Reluctant to interact with engineers from other disciplines
6	Different mental models of the system, task and design related phenomena
7	Lack of a common language to discuss freely at creative meetings
8	Education within disciplines does not call for integration in professional life
9	Knowledge transfer between domains is inadequate
10	System engineers are lacking detailed information of the system

For three of the challenges listed in Table 2.3, a detailed explanation of how the state of the art currently addresses these challenges is offered. Special attention is payed to how they are represented in the earlier stages of systems engineering.

Common understanding of overall system design

Stakeholders from various domains each have their own models and approaches to describe and understand a system. When considering communication support that includes non-engineering stakeholders, three main directions are identified. First, approaches based on functional modeling [Erden *et al.*, 2008] can be used to describe the system at an abstract level. Especially in the early stages, this abstract level allows stakeholders from all domains to reason about the system appropriately. Second, this can be achieved by using virtual reality and visualization applications [Korfiatis *et al.*, 2012, Madni *et al.*, 2014, Thalen, 2014]. The third option is to condense architecting information into a single and accessible overview. This is for example done in the A3 Architecture Overview method [Borches, 2010]. The combination of multiple views and support of visual aids allow all stakeholders to understand the system.

Transfer of models and information between domains

Opposed to providing one accessible view to all stakeholders, model transformations aim to connect viewpoints by transforming one viewpoint into another. Sometimes an intermediate representation is used, such as SysML in [Shah *et al.*, 2010]. Another option is multi-domain simulation such as Modelica [Modelica Association, 2012] or the bond-graph based 20-sim [Controllab Products B.V., 2011]. This is effective, but has as drawback that it is heavily focused on the physical domains, which are often less relevant in early stages of systems engineering. It could however be used to also perform high-level synthesis, as is described in [Canedo and Richter, 2014]. [Alvarez Cabrera, 2011] states:

"Useful descriptions include information from different sources and different characteristics. Just gathering the common information in its original format (i.e., domain-specific models) for a discussion is not enough because, without a clear association among the domain-spe-

cific concepts, it would require that each stakeholder understands the specific language of every other involved stakeholder"

This means that when multiple views are presented at the same time, they are only useful if there is a clear association between the various views.

Lack of a common language to support creative meetings

A group of diverse stakeholders cannot reason together about a system easily, as there is a lack of a common language. SysML has been proposed as this common language but is less accessible for non-engineering stakeholders [Madni *et al.*, 2014]. In [Adamsson, 2005] a practical approach is suggested. Co-location of key stakeholders could influence collaboration and communication positively. [Rozanski and Woods, 2012, Torry-Smith *et al.*, 2011] conclude that while a common language is needed, this does not seem to be feasible as different views are always needed to describe a system.

Additional Needs

In addition to the comprehensive review of [Torry-Smith *et al.*, 2011], earlier work of the author [Haveman, 2009] has identified needs of system architects through a series of interviews with system architects at several companies developing complex systems (radar systems, medical devices, printers, consumer products). Here, eleven needs were identified of which five were classified as communication needs. These are listed in Table 2.4. While some of those are reflected in the challenges identified by [Torry-Smith *et al.*, 2011], the second need mentions an interesting aspect, namely the desire to reduce irrelevant information while still delivering the right information. [Borches, 2010] applied this concept in the A3 Architecture Overview method, by restricting the presented information to two sides of an A3 paper.

2.3.3 Conclusions

This section has given an overview of relevant communication models and also identified existing communication challenges in literature. Challenges that are considered relevant for the research presented in this thesis are a common understanding of the overall system design, supporting transfer of models and information between domains and a lack of a common language to support creative meetings. Supporting the communication process will be addressed in Chapter 4. Finally, an additional need expressed by system architects is to reduce the irrelevant part of information during communication.

Table 2.4 – Identified system architect needs related to communication [Haveman, 2009]

1	Keeping a structured overview on what has been communicated with a certain stakeholder
2	Deliver the right information to stakeholders while keeping the irrelevant part of information low
3	Make unavailable persons (due to busy schedule or being in a different location) available for the optimal form of communication
4	Make sure that information is conveyed and interpreted correct
5	To hand over project information as complete as possible

2.4 Decision Making in Conceptual System Design

A very important concept in conceptual system design is decision making. Each design methodology that is described in section 2.1 relies on decisions to move forward in the described process. These decisions range from whether a problem has been explored enough to whether a solution is satisfactory. During conceptual system design, not every detail is known yet. Therefore decision making will always be done under uncertainty [Muller, 2004, p56]. According to [Neumann and Morgenstern, 1944], rational actors will always select the optimal choice, given that there is no uncertainty. However, this theoretical situation is difficult to achieve. First of all, there is always some measure of uncertainty. Second, the optimal option for one stakeholder might not be the optimal option for another. Therefore, this section explores how decision making, uncertainty and the concept of an optimal decision are related.

2.4.1 Theory

Decision making is based on two types of theories, normative and descriptive [Hansson, 1994]. Normative theories focus on how decisions should be made. An example of this is that human decision making is known to follow a bounded rationality [March, 1978]. Descriptive theories focus on how decisions are actually made, for example using heuristics with known biases. [Tversky and Kahneman, 1974] describe three heuristics that are commonly employed in as they call it, making judgments under uncertainty. These are representativeness, availability and adjustment. These heuristics are able to influence the rationality, even if decision makers are experts in their field. For example the adjustment heuristic states that the final assessment of a value is influenced by its initial assessment.

To apply decision theory to conceptual system design, utility theory [Neumann and Morgenstern, 1944] can be used as a normative decision model in the sense that it is possible to establish a scalar function, termed a utility function, over the domain of attribute values. This is further detailed in [Malak *et al.*, 2009]. They state that uncertainty can be either variability or imprecision. Variability is naturally random behavior which can be predicted by a probability density function. Imprecision is caused by a lack of knowledge. The concept of value assumes that maximizing the utility of a system will lead to the highest value, and that should be strived for in a design [Lee and Paredis, 2014]. A rational decision process will always choose the alternative with the highest utility for a system, in order to maximize its value. When groups of individuals are involved, the concept of value will not be equal across all stakeholders. In this case, each involved individual could try to maximize the value of the overall system, but this is not a rational process. Therefore it is better to assume that stakeholders will aim to maximize their own value [Lee and Paredis, 2014].

Criticisms of decision theory include the ludic fallacy [Taleb, 2007]. This criticizes the application of game theory to real life situations because it uses structured randomness to model life's unstructured randomness. Taleb argues that it is impossible to be in possession of the entirety of available information. "Known unknowns" are represented by the concept of uncertainty in [Malak *et al.*, 2009] or by the concept of time-independent real complexity

[Suh, 2005]. However, "unknown unknowns" also exist. Examples of these include small details that influence the eventual system in a meaningful way that could not have been predicted, or the inability to predict events that were previously unobserved. An example of this would be that stock market models never included the effect of the terrorist attacks on the twin towers on the 11th of September 2001. [Suh, 2005] refers to this as time-dependent combinatorial complexity.

A final aspect that must be treated here is the completion of the conceptual system design phase. Clearly, this is a decision as well. This decision can be either time or budget bound, but on a higher level it entails the notion that the conceptual system design is sufficiently appropriate to continue with the detailed system design. However, a tendency to continue with an analysis process exists, because for example value X is still uncertain, or the behavior of component Y is still not quite understood. To decide whether or not to continue the refinement, a trade-off analysis must be made. This can be done based on the value-of-information theory [Howard, 1966]. This theory aims to place a value on the reduction of uncertainty. [Thompson and Paredis, 2010] discuss the application of this theory to conceptual system design and show that it is possible to quantitatively determine whether or not an analysis process should be performed.

2.4.2 Practice

Some applications of decision theory have already been alluded to in the previous sections. However, some more will be treated here. A prime example of the application of decision theory can be found in the area of multi criteria decision making techniques.

Two approaches can be chosen here, that either consider a list of alternative solutions, or consider a solution set. This solution set is based on set-based concurrent engineering [Sobek *et al.*, 1999], which has been discussed earlier. [Malak *et al.*, 2009] discuss set-based design in the light of utility theory and show that utility-based decisions allow delayed commitments to a particular solution while still making rational decisions.

[Mavris, 2009] lists a large number of multi-attribute decision making techniques. He concludes that the selection of the final concept is heavily dependent on the subjective rankings assigned to the customer importance, especially when using design methods such as QFD [Mizuno and Akao, 1994]. Furthermore, he identifies three challenges, which are incomplete information, requirements uncertainty and the actual selection process. The use of surrogate models to approximate system elements is suggested. Surrogate models can be considered as low-order models, which are often used by system designers as identified by [Bonnema, 2008]. Using these kind of models in a decision making process is described by [Muller, 2004] and can be seen in Figure 2.6.

In order to support the decision making process, knowledge must be collected, conveyed and communicated. [Grogan, 2014] adds to this that as decision-makers are seldom the same people performing analysis and design activities, there is a challenge in communicating insights from modeling activities to overcome the normal course of decisions. In the civil engineering community, particularly in urban planning and infrastructure, much effort has been spent in this area already and this need is recognized. For example [Hansman *et al.*, 2006] write:

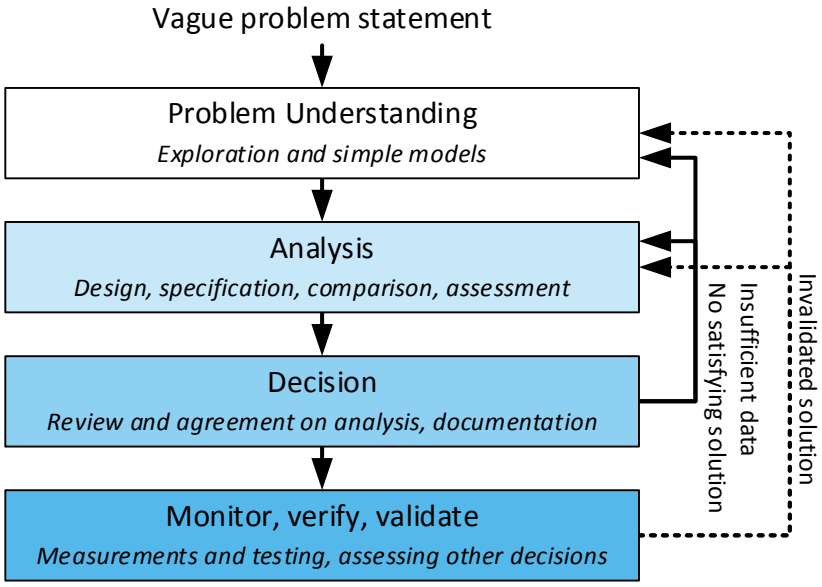


Figure 2.6 – CAFCR Decision making approach - flow from problem to solution. Reproduced from [Muller, 2004, p55]

"Decision makers and other stakeholders need means for seeing, experiencing, and experimenting with infrastructure alternatives and their implications" and *"a major research effort should be to develop methodologies to increase effective communication, understanding and alignment among stakeholders with diverse backgrounds, objectives, levels of technical understanding".*

An example application by [Arias *et al.*, 2000] aims to create shared understanding through collaborative design by implementing an Envisionment and Discovery Collaboratory. This combines physical objects, simulations and drawings in a participatory design environment. Another example with a healthcare application is given by [Garde, 2013], who focuses on the use of physical objects in a participatory game environment.

In systems engineering, less work on this topic has been realized, with a few exceptions. [Lu *et al.*, 2007] state that new approaches are required that go beyond current work in concurrent engineering to emphasize collaboration. They for example emphasize the need for strategic guidance to construct common understanding. Example applications can be found in [Grogan *et al.*, 2015] who focus on interactive models in an online database, [Grogan and Weck, 2015] who focus on the use of gaming and [Mavris *et al.*, 2005] who focus on visualization technology by providing a large-screen visualization environment.

[Heer and Shneiderman, 2012] provide some more considerations for collaborative visual analytics. They start with the premise that sensemaking is often a social process involving parallelization of effort, discussion, and consensus building. Next to this, they emphasize the need for common ground (see also section 2.3.1), as well as incentives and engagement of the stakeholders. Their research concludes with a selection of design considerations, such as the support of pointing techniques, establishing personal relevance of data sets in order to increase engagement and group diversity which can result in more complete results.

2.4.3 Conclusions

The discussion on decision making has provided an overview on the theoretical basis of decision making. In its application, the research presented focused on the fact that decision making in conceptual system design is a group effort (although decisions are made on a personal basis as well) and shows the possibilities of various techniques that support this decision making process.

2.5 Models and Simulations

Throughout this chapter, and also in the introduction, models and simulations have been mentioned numerous times already. This is not surprising, as especially models are an integral tool of conceptual system design. They are the principal language of an architect [Maier and Rechtin, 2000] and allow a designer to abstract from reality [Bonnema, 2008]. Therefore they serve an important purpose in conceptual system design, as abstraction is required here. Multiple definitions of models exist that are fairly similar. Most definitions mention representations, systems aspects and the system itself. In this thesis, the same definition is used as in [Maier and Rechtin, 2000] as the characteristics that are explicitly mentioned align to this research. The definition used by [Maier and Rechtin, 2000] is given in [IEEE, 1990] and states that:

A model is an approximation, representation, or idealization of the structure, behavior, operations, or other characteristics of a real-world process, concept or system.

The following definition is used for a simulation, based on the definition given in [National Research Council, 2002]. This is similar to the definition given in [Walden *et al.*, 2015].

A simulation is the implementation of a model in executable form or the execution of a model over time.

[Walden *et al.*, 2015] add to this that simulations provide a means for analyzing complex dynamic behavior of systems, software, hardware, persons and physical phenomena. Note that according to this definition a simulation does not necessarily have to be implemented in a computer tool, it simply is executing a model over time (see also section 4.3.2). A thought experiment can be termed a simulation as well, as it considers a model in various instances. An example in this regard is using soft systems methodology via systemigrams [Boardman and Sauser, 2008].

In the rest of this section, various aspects of models and simulations are treated. First of all, the role of models in design is discussed, especially in the light of model-driven design. Second, several general modeling methods and methodologies are described. Finally several model types are highlighted. [Maier and Rechtin, 2000] classify models by view and distinguish six types of models:

- Purpose/Objective; models supporting definition of purpose and objectives
- Form; scale models or block diagrams of system describing physical aspects
- Behavior; models detailing the behavior of the system

- Managerial; models describing the development process
- Data; models describing usage and relations of data in system
- Performance; models describing the performance of the system

While all of these model types are interesting to discuss further, the focus in this research is on purpose/objective models, because the conceptual system design stage aims to represent this in a design, and behavioral models, as the interest lies in supporting these better in the conceptual system design stage. Performance models are also interesting, but [Maier and Rechtin, 2000] do not provide examples, and only categorize performance models in analytical (e.g. budgets), simulation and judgmental (e.g. expert estimations) models. Therefore, examples of performance models are given in the two categories that are discussed in more detail.

2.5.1 Model Driven Design

In recent years, a need has been identified to make a paradigm shift from document-centric to model-centric systems engineering [Robinson *et al.*, 2014]. Using models as a primary means of information exchange instead of a document-centric approach can improve consistency, re-use and integration [Woestenenk, 2014]. One of the most well-known formal approaches to using models in systems engineering is Model Based Systems Engineering (MBSE). [Bonnema, 2008] researched which models are used in conceptual system design through a survey with system designers. These include, but are not limited to, budgets, sketches, scenarios, functional diagrams, block diagrams and mathematical models. However, the usage of models does not make it model driven design directly. Model driven design requires something more, which is defined well by [Estefan, 2008]. It characterizes MBSE and model driven design in general as:

"In a nutshell, model-based engineering 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"

This is especially difficult in the conceptual stages of design, as capturing information in formal models to be re-used in later design stages has traditionally not been the focus when creating models in this stage. The Model Based Conceptual Design (MBCD) initiative [INCOSE, 2013] focuses on the conceptual design phase and aims to transform artefacts produced in these early stages to representations that can be used in the MBSE approach. Furthermore, there still is a key need to understand how to apply a model-based approach in these conceptual design stages [Sampson and Friedenthal, 2014].

A recent survey by the MBCD working group aimed to identify existing challenges for professionals involved in conceptual system design [Morris *et al.*, 2015]. The respondents include practitioners, managers and decision makers mainly from aerospace and defense industries. Table 2.5 shows the most important identified challenges, but also reported successes, both sorted by the number of mentions. Stakeholders were also asked to report how challenges were affected due to MBSE. Interestingly, respondents that identified the challenge "lack of stopping criteria" indicated that MBSE made this issue worse. Further-

Table 2.5 – Top Challenges and Successes in MBCD Survey [Morris et al., 2015]

Challenges
Lack of best practices and "Return on Investment" information
Lack of a specific methodology/description
Lack of stakeholder engagement
Lack of "stopping criteria" for modeling
Solutioneering (Focusing on solutions instead of the problem)

Successes
A clearer understanding of the problem
Informing requirements development
Good stakeholder engagement
Integration of Modeling and Simulation

more, stakeholder engagement was identified as both a success and a challenge but more detail as to why this occurred was not offered.

During an interview with a system architect at Philips Healthcare (see Appendix A, interview 3), the architect remarked that *"MBSE is often too rigid and requires too much. A lighter MBSE approach is hard to attain, but exceedingly valuable"*. The Boderc design methodology [Heemels *et al.*, 2006b] propagates the use of simple models to identify issues and could thus be worthwhile in this regard.

2.5.2 General Modeling Methods and Methodologies

This section discusses modeling methods and methodologies. These include methodologies that are considered to be MBSE methodologies, functional modeling approaches and a few other approaches. Some of these methods have already been mentioned in previous sections.

In [Estefan, 2008] and [Reichwein and Paredis, 2011] overviews are given of various Model Based Systems Engineering methodologies, such as IBM Rational and the INCOSE Object-Oriented Systems Engineering Method. These methodologies use the visual modeling language UML (Unified Modeling Language) and SysML (Systems Modeling Language) [Object Management Group Inc., 2012]. SysML is an extension of UML and is created especially to model complex systems, while UML is oriented more towards software engineering. SysML is domain independent and aims to foster communication between disciplines. However, the accessibility of SysML for non-engineering stakeholders is questionable [Madni *et al.*, 2014].

Another example of the application of MBSE is the Architecture Modeling (AM) language of [Woestenenk, 2014], which is an approach that tries to capture all relevant engineering information during complex systems development. Differences with SysML are that the AM models different views concurrently. Furthermore, the AM language avoids encapsulation, with the argumentation that the elements that are to be encapsulated can contain important design decisions in their own right and should not be part of a "parent object"

A final approach that is considered by [Estefan, 2008] to also be a MBSE methodology is the Object-Process Methodology (OPM) [Dori, 2002] which is based on the functional, structural and behavioral aspects of a system. Using a single diagram type, Object Process Diagrams (OPD), a systems structure can be described together with behavioral aspects. In

[Yaroker *et al.*, 2013], it is stated that the fact that modeling languages exhibit a high level of abstraction results in ambiguous elements and messages. However, they argue that a model-based executable simulation can resolve these issues and present OPCAT, an execution environment for OPD models. Experiments with this tooling suggest that users are better able to understand and reason about a systems' dynamics. However, they also paid less attention the systems structure compared to using non-executable OPD diagrams.

A commonly used modeling methodology in conceptual system design is functional modeling. [Erden *et al.*, 2008] discuss various approaches and applications. They define functional modeling as:

"the activity of developing models of devices, products, objects, and processes based on their functionalities and the functionalities of their subcomponents".

Furthermore, [Erden *et al.*, 2008] argue that functional modeling can bridge the gap between system level design and detailed design, by supporting multidisciplinary design. [Woestenenk, 2014, p50] states that functional modeling is interesting because the functions a system performs are often more complex than the system's physical artefact. [Erden *et al.*, 2008] also conclude that functional modeling can act as a framework for computer reasoning. In recent work, [Canedo and Richter, 2014] discuss a computer reasoning implementation. This enables architectural design space exploration using a functional modeling compiler, which is connected with physics based simulations such as Modelica [Modelica Association, 2012].

The Octopus toolset [Basten *et al.*, 2010] allows model-driven design space exploration, based on the aforementioned Boderc design methodology [Heemels *et al.*, 2006b]. [Basten *et al.*, 2010] state that the Octopus toolset supports high-level modeling based on a clear separation of application, platform and the application-to-platform mapping. This modeling approach is the Y-chart methodology originally developed by [Kienhuis *et al.*, 1997] for programmable embedded systems. The Octopus toolset connects different tools through an intermediate model representation and uses domain-specific abstractions to support different application domains. This ensures reuse of the model across domains and allows both performance analysis and formal verification based on derivations of the same model.

A final approach is the Design Framework [Moneva *et al.*, 2011]. This framework provides a mechanism to use heterogeneous models for different system elements and links them using design parameters. In this way, the Design Framework allows for conceptual system design that uses simple models, and offers a MBSE framework that allows a connection of these models with the overall system knowledge database.

2.5.3 Objectives & Purpose Models

Objectives and purpose models are those that try to formalize user or business needs and relate these to high-level system information, such as functions, use cases or other parts of the system architecture. The Quality Function Deployment model [Mizuno and Akao, 1994], described in [Blanchard and Fabrycky, 2010], constructs one or more matrices to do so. The most well-known one is the House of Quality. This matrix relates user needs to design attributes and important design parameters. An approach that also relates various

aspects via a matrix is FunKey architecting [Bonnema, 2008]. In FunKey architecting, functions and key drivers (generalized stakeholder needs) are related. This is done by identifying whether a function influences a key driver and whether this contribution is positive or negative. This can be extended to initial budgets for the system. Modeling the systems context can be done by making use of systemigrams [Boardman and Sauser, 2008] or pictorial influence diagrams [Bonnema *et al.*, 2015].

[Aguwa *et al.*, 2012] propose a method that makes "the voice of the customer" explicit in a quantitative format and ties it to product information. Another modeling method that tries to capture and validate user requirements is proposed by [Beek and Tomiyama, 2011]. This method uses high-level understandable and decomposable workflow models.

Finally, research on using formal models when specifying requirements to avoid ambiguity has resulted in a large body of work. Examples are [Braun *et al.*, 2010] in which a guideline and a tool for requirements specification, REMsES is introduced, or [Liu *et al.*, 2012] in which requirements are captured using a scenario-based approach. Interesting research results can be found in [Sikora *et al.*, 2011], which among others states that requirements models should support specification of requirements across different abstraction levels.

2.5.4 Behavioral Models

Behavioral models refer most often to lower level executable formal models that model certain aspects or behavior of the system. [Maier and Rechtin, 2000] also coin this term and describe them as models of what the system does, opposed to models of form (structure) which describe what the system is. Table 2.6 gives an overview of the various model types identified by [Maier and Rechtin, 2000]. In addition, a short description and the main purpose are given. In the following text these model types are discussed in more detail.

Table 2.6 – Overview of behavior model types identified by [Maier and Rechtin, 2000]

Behavior Model Type	Description	Main Purpose
Threads and scenarios	Also termed use case, these are a sequence of system operations. An ordered list of events and actions.	Client communication, builder communication, design reviews with respect to behavior [Maier and Rechtin, 2000]
Data and event flow networks	Define the behavior of a system by a network of functions or processes that exchange data objects	Collapse threads into compact and more usable models [Maier and Rechtin, 2000]. Overview of the whole system with terms that are easy to comprehend. Interactions in lower domains are clarified [Erden *et al.*, 2008]
Mathematical systems theory	Multidimensional feedback systems which can either be continuous, sampled (temporal discrete) and discrete event	Understanding feedback operations in systems [Maier and Rechtin, 2000]. Analyze the behavior of multi-domain dynamic systems [Controllab Products B.V., 2011]
Autonomous agent, chaotic systems	Model of interaction between multiple components or systems, which can be of the same type	Model interactions between system elements [Maier and Rechtin, 2000]. Identify emergent behavior [Baldwin *et al.*, 2015]
Public choice and behavior models	Models of human behavior and decision making	Understanding the human system [Maier and Rechtin, 2000]

Considering the threads and scenario type, especially scenarios are useful tools to envisage future usage scenarios and contexts for the system [Ionita, 2005]. More recently, [Curry and Ross, 2015] discuss extensions to the Epoch-Era Analysis framework. This is a framework designed to clarify the effects of changing contexts over time on the perceived value of a system in a structured way and is based on research presented in amongst others [Ross, 2006].

Data and event flow networks are described with notations that are either based on data flow diagrams or functional flow block diagrams [Maier and Rechtin, 2000]. These decompose the systems functions in a hierarchical way. Functional modeling is reviewed in [Erden *et al.*, 2008] and for examples identifies the use of function-behavior-structure [Gero, 1990]. They discuss that some of the behaviors observed with objects can be considered as functions on a higher abstraction level. In their function-behavior-state model [Umeda *et al.*, 1996] define a function as a description of behavior recognized by a human through abstraction in order to utilize it. Furthermore, they define behavior as "*sequential state transitions along time*".

Mathematical systems theory is a very wide area that touches on many of the core domains of science and engineering. Examples of mathematical models are often found in physics-based modeling applications. For example Modelica [Modelica Association, 2012] which is widely supported throughout a large part of the mechatronics development community. Another example is 20-sim [Controllab Products B.V., 2011], which uses bond-graphs as a modeling paradigm. However, discrete event approaches are commonplace as well. A foundation of discrete event modeling is the Discrete Event System Specification [Zeigler *et al.*, 2000]. Two other examples are Action State Diagrams [Kuuluvainen *et al.*, 1991] and Software Hardware Engineering (SHE) [Theelen *et al.*, 2007]. Within SHE, the language used for formal model specifications is the Parallel Object-Oriented Specification Language (POOSL) [Putten and Voeten, 1997].

Agent based modeling is used to model interactions between various system types. This often allows modeling of emergent behavior. This is especially useful when modeling Systems of Systems [Baldwin *et al.*, 2015].

Public choice models are for example focus groups and multiple user simulations [Maier and Rechtin, 2000] which also have been discussed in section 2.4.2 [Arias *et al.*, 2000, Grogan, 2014]. The outcomes of these models can be used as design input, but observation of how the results were created helps to understand human behavior and decision making, which can also be used as design input.

2.5.5 Conclusions

This section has given a definition of models and simulations and an overview of high level models and lower level models that are often executable. It has highlighted that MBSE or similar model-driven design variants are gaining more and more traction in the systems engineering community. However, multiple challenges still exist, such as the lack of support to apply model driven design in the conceptual system design stage. Finally, especially the discussion of the various types of behavioral models showed that there is a wide array of representations to describe behavior of complex systems.

2.6 Problem Definition

Based on the research motivation given in section 1.4 and the review of literature in this chapter, this section aims to define the problem that will be addressed in this research. It does so by summarizing the literature review and presenting the research questions.

Supporting multidisciplinary discussions on the behavior of systems during their design is stated as the main goal in the research motivation. The assumption is that it is important for all stakeholders to have insight in and be able to discuss key characteristics that influence the system behavior. This should ultimately contribute to a better design.

It is interesting to observe how system architecture frameworks deal with the behavior of systems. The 4+1 method by [Kruchten, 1995] integrates behavioral and structural views of the system by considering behavior over these structural views. This means that in order to understand the behavior of systems, all structural views must be observed as well. This seems to conflict with an identified system architect need, which is to *"deliver the right information to stakeholders while keeping the irrelevant part of the information low"*. The question is how to optimally analyze system behavior and how this can be done in a problem focused manner, which was shown to be important in conceptual system design.

In order to analyze system behavior, models and simulations are crucial tools. However, communication of models and simulations is not straightforward, and while efforts have been made to do so, [Lu *et al.*, 2007] state that new approaches are required that allow collaborative design. Furthermore, [Morris *et al.*, 2015] identified that there is for example still a lack of support in the conceptual system design process.

Finally, research in this thesis will focus on engineered systems, and not on systems of systems. This choice was made to have a manageable scope of the research. It also aligns with the activities of both the research group and project in which this research is situated.

These considerations lead to the following research questions:

RQ1: What is the role of behavioral analysis in the conceptual system design process and which views are required to represent behavior?

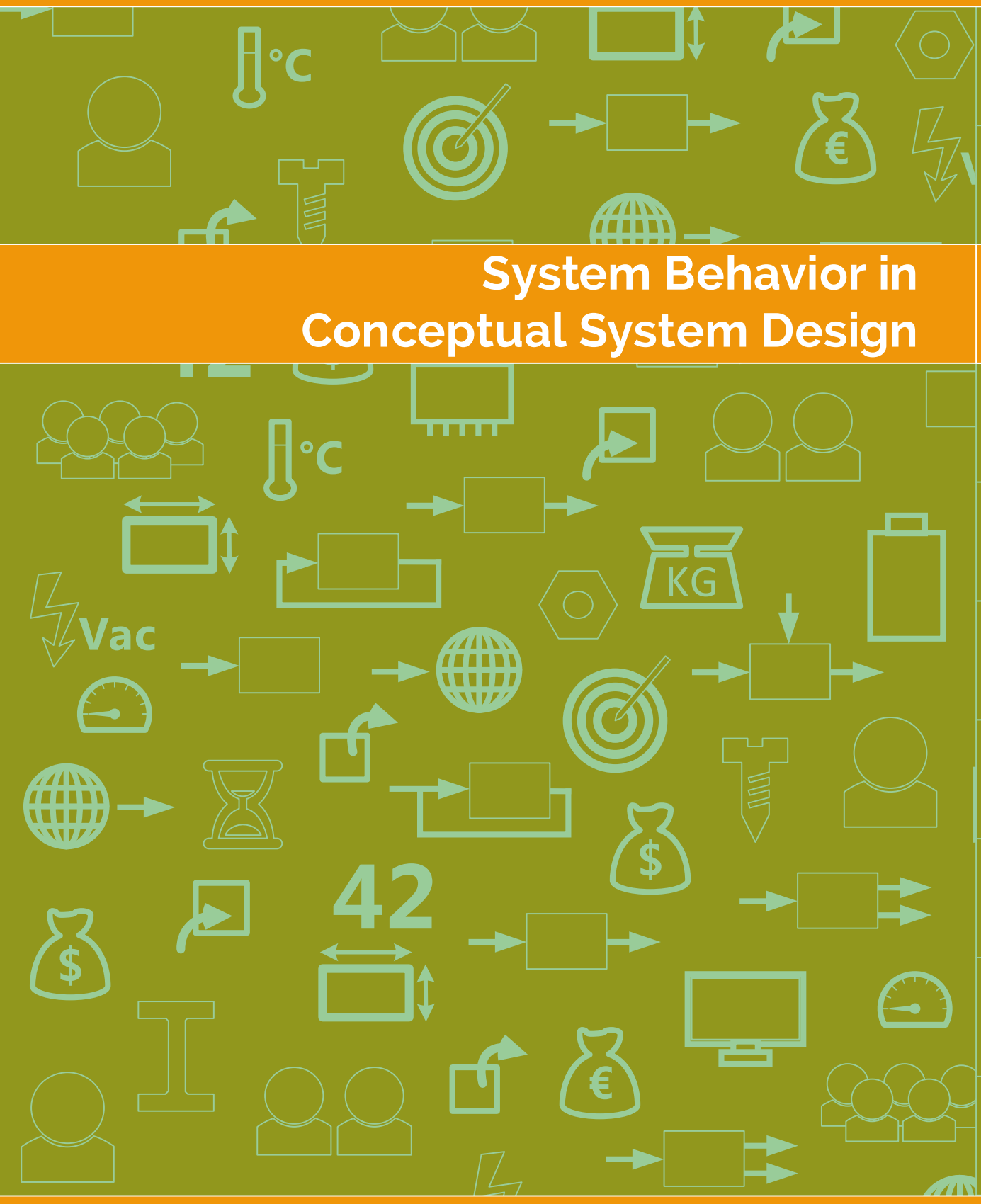
RQ2: What are the main challenges that oppose effective communication of simulations in the conceptual system design stage?

RQ3: How are important concepts in communication of models and simulations represented in existing architecture frameworks?

RQ4: What does a method that supports communication of models and simulations in conceptual system design look like?

RQ5: What is the applicability and value of the method resulting from RQ4?

Chapter 3 will explore RQ1 and answer this question based on observations and interviews in industry, as well as a more in-depth literature review. Chapter 4 aims to answer both RQ2 and RQ3, using the same strategy as in Chapter 3. RQ4 will be answered in both Chapter 5 and Chapter 6 through case studies and subsequent method development, while Chapter 7 and Chapter 8 focus on answering RQ5 by evaluating the proposed method.



System Behavior in Conceptual System Design

3

System Behavior in Conceptual System Design

This chapter analyses in which manner consideration of system behavior should be included in the conceptual system design process. The conceptual system design process is revisited with a discussion of this process in industry in section 3.1, including the role of behavioral analyses in conceptual system design in section 3.2. Section 3.3 proposes a practical conceptual system architecture that allows inclusion of system behavior in the conceptual system design stage.

Data sources for this and the next chapter were in-depth literature reviews as well as observations and interviews in industry. Eleven interviews were held with system architects or persons in similar roles. Six of those were held at the interventional X-ray division of Philips Healthcare, while five took place at companies operating in the logistics, space and optics domain. The interviews generally occupied one to two hours. On one occasion a follow-up interview was scheduled. Interview questions were prepared upfront each time, but adapted for the occasion. The questions were mainly used as a tool to structure the free-flow interviews. Therefore no formal report on the interview results is given, but they are summarized in Appendix A. Furthermore, as the Allegio project, in which this research is situated, used an industry-as-laboratory approach, the researcher has been embedded in the organization of Philips Healthcare for the full duration of this research project. Therefore, this is the context when referring to observations made in industry.

3.1 The Conceptual System Design Process Revisited

This section takes another in depth look at the conceptual system design process. It does so by identifying key objectives, classifying types of design processes and identifying ways to approach the conceptual system design process from a practice-oriented view.

3.1.1 Types of Conceptual System Design Processes

One of the goals of the interviews and observations in industry was to identify the types of conceptual system design processes. In many cases, multiple designs are necessary to meet different sets of stakeholder requirements (see interview 1, Appendix A). Although a conceptual system design process seems to imply that a completely new system is being designed, most often this is not the case. It could be that a new subsystem is introduced, for example a robotic control system for catheters that needs to connect with interventional X-ray machines. However, more often subsystems will be updated to replace legacy hardware or software and to add functionality. A completely new system is a rare occurrence. It is sometimes considered to occur in domains such as space applications, although also in this domain re-use of existing system parts plays a large role as well to

meet reliability requirements (see interview 11, Appendix A). This can also be observed in conceptual system design study reports [ESA, 2014].

In order to categorize the various types of design processes that were observed in industry, Figure 3.1 was developed, based on [Haveman and Bonnema, 2013]. The figure represents various types of information. First of all, it lists four categories of design processes. These categories focus on the type of process and less on various stages in the innovation process (see interview 4, Appendix A). Second, it lists common knowledge sources that are used during that specific design process. Third, it categorizes these knowledge sources on two axes, cost and accuracy. These axes were chosen as cost is the most important driver to choose analysis methods in industry, while accuracy gives an indication of the value of an analysis method with respect to the information gained. Finally, within each design process a model is represented that envisions the model that should support communication of behavior in these design processes. The positioning of this "communication model" is intentional as it is aimed to be a relatively low-cost, low-accuracy model, but with a higher accuracy than basic estimations.

The knowledge sources that were identified in the conceptual system design process are the following:

Basic Estimations; these are low-order models and calculations that are based on assumptions of domain experts. They are usually the initial assessment of a problem. It could even be a small sketch on the back of an envelope.

Documentation; this is the explicit knowledge database that has been composed by an organization. It describes for example system requirements specifications (see interview 2, Appendix A) or system design specifications of previous products or versions.

Single Aspect Simulation; these are simulations that determine the performance of a single aspect. The scope of these simulations is usually confined to the quantification of a key driver within a single subsystem. In industry, few system-wide simulations have been observed. This could be due to the fact that the organizations where the observations took place had an organizational structure that reflected these subsystems [Conway, 1968], with only system architects responsible for the system wide overview.

Measure Old System; this includes performance measurements or behavioral observations of the old system. This can serve as additional information next to existing documentation, because often not all behavior is documented completely. It was observed that somewhat ambiguous or unclear behavioral requirements in documentation were verified on old systems during the conceptual system design phase. Often the choice was made to use the interpretation of the requirement as it was observed in the old system.

Measure Prototype or Functional Model; if a subsystem is redesigned, it is often possible to develop functional prototypes or models quickly to experience how they behave (see interview 8, Appendix A). This category refers to actual implementations that are either in the mechanical domain as prototypes or in the software domain as implemented functional models. An example given by an interviewee was they run new software in a development

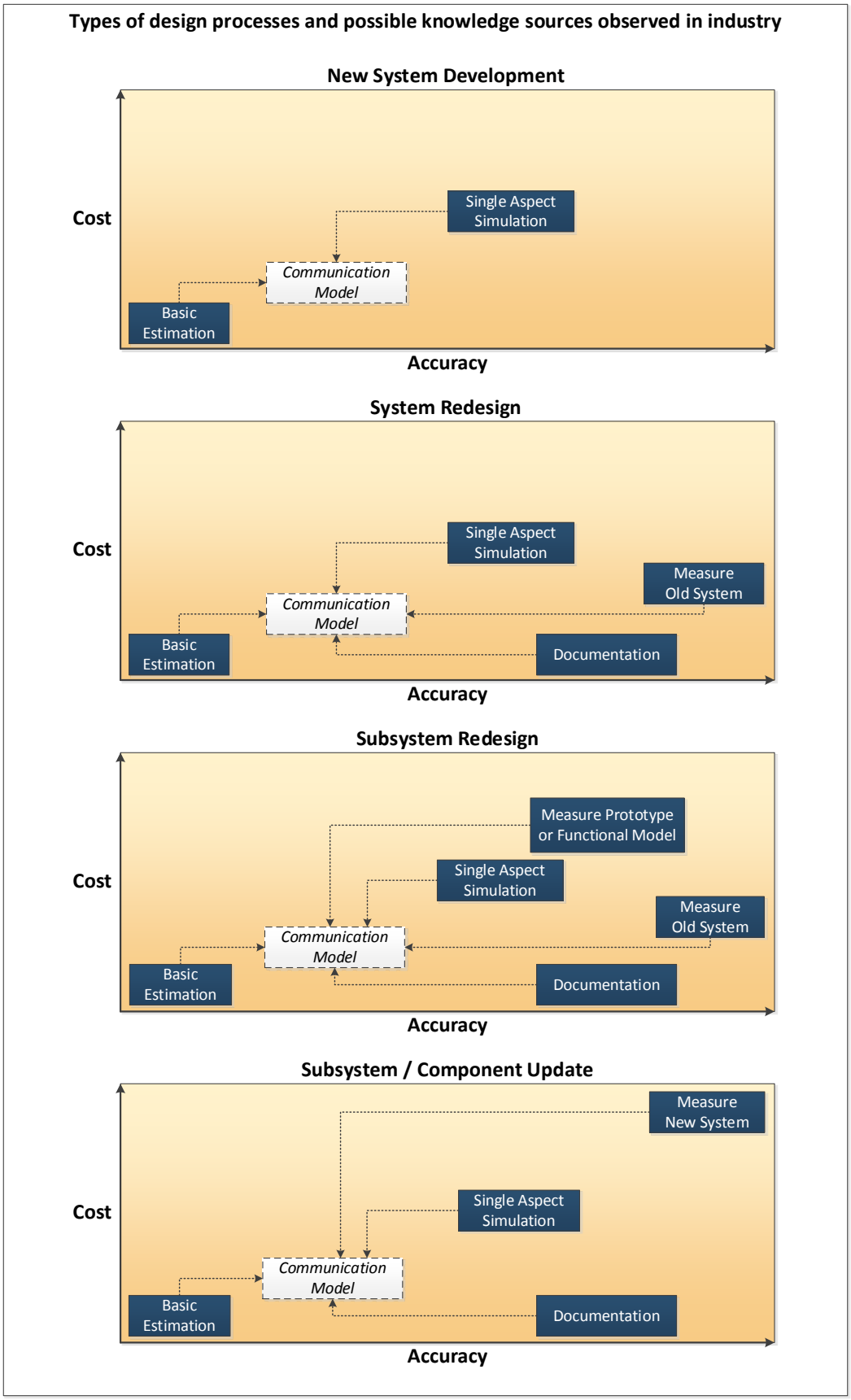


Figure 3.1 – Types of design processes and knowledge sources in conceptual system design

environment that does not completely correspond to the final system environment. In this way, valuable results can still be gathered and it is faster than waiting for the implementation to be developed.

Measure New System; In addition to measuring prototypes or functional models (with small changes) it is possible to test the new system directly. An example given by an interviewee was that when migration to a new computer architecture was required, this new computer architecture was often already available (see interview 5, Appendix A). This stems from the fact that companies tend to stick to components until new components have proven their reliability elsewhere, or even until the old component is not available anymore. By acquiring the new computer architecture and transferring the existing software to that system, the new system is almost completely implemented and can be observed and measured once more.

The types of design processes that were identified in the conceptual system design process are the following:

New system; developing a completely new system is possible, but as discussed before, it is not common. Components of old systems will always be re-used in a new system. Clearly, there is a grey area between when a system can be considered a new system and when it is considered to be a redesign. A good measure for this would be to see if the main system goal is different than from previous systems. Available information sources are basic estimations, and single aspect simulations that are based on the re-use of existing components.

System redesign; a system redesign can rely on measurements of old system versions to develop assumptions and validate expectations. Also, a wide documentation base should be available compared to development of a new system. A system redesign will be so impactful that multiple subsystems of a system will be significantly changed after the redesign.

Subsystem redesign; a subsystem redesign considers the same things as a system redesign, but the major changes to a system will be contained within a single system, though ripple effects might occur that require other subsystems to be changed as well. This is the most common type of redesign process, though it has been observed that often multiple of these processes are combined in a single project that aims for a new system release.

Subsystem / component update; the last type is a subsystem or component update. Due to the fact that this is only a small update, it is often possible to test the new system right away in the actual system in operation. The experiment can take place in a development setting. Think of a test track for cars or a recreation of an exam room for medical devices. The difference with a subsystem redesign is that a subsystem update generally does not focus on the problem domain, but implements a new solution straightaway.

A communication model as envisioned can be useful in all cases. It will however be more valuable if there is no new system or functional model to be used as a prototype.

Table 3.1 – Key drivers (left) and subsystems (right) of an interventional X-ray machine

Key Drivers
System Cost
Total Procedure Time
Patient Health/Safety
Image Quality
Hand-Eye Coordination
Staff Safety
3rd Party Integration

Subsystems
Patient & Beam Positioning
X-ray Generation
Image Processing
User Control
Image Display
Power Distribution

3.1.2 Common Approaches to Conceptual System Design

During observations and interviews it was identified that there are three ways to approach the conceptual system design. It must be noted that these are not mutually exclusive. To clarify these approaches, an interventional X-ray machine will be used as an example. This system has a number of key drivers (see section 2.2.1 for an explanation of key drivers) and several subsystems. These are listed in Table 3.1.

The first way to look at the problem is to consider the system as a whole and analyze how functions are allocated over subsystems, as well as the influences of the systems context and needs that stem from this context. In an interview, a value engineer indicated to reconsider the iXR system for different types of applications, such as for example redesigning a high-end system to a cheaper variant that exhibits less functionality. In this way, the system is assessed as a whole which allows system designers to determine if other functional variants are possible that perform better in different contexts (see interview 1, Appendix A). For example, an iXR machine could be a lot less complex and cheaper if the function of patient positioning is fulfilled by the user instead of the system. When reviewing the portfolio of interventional X-ray systems offered by Philips, it can be observed that these kind of systems are indeed being offered [Royal Philips, 2015b].

The second way to consider a system is from a key driver perspective. This means that a key driver of the system is selected and analyzed across the system and its subsystems. This approach is often employed by system architects that are responsible for an overarching aspect of the system, as larger systems often have a team of system architects to manage the architecture of the system. Conceptual studies towards improving these aspects are often performed outside of the structure of projects and do not aim for direct implementation. An example can be given for reasoning from the key driver "procedure time". In this case, the system is considered as a whole from various viewpoints such as a functional viewpoint. The analysis focuses on the impact of elements in these viewpoints on the key driver procedure time. This allows discussions on where time savings might be possible. As a note, considering one single key driver does not mean that the impact on other key drivers should be forgotten.

The third approach is to look at the system from a subsystem perspective. In organizations it is common to focus design processes on subsystems, to ensure that the actual work

is contained within an organizational unit as the organizational structure will often reflect the system structure [Conway, 1968]. As an example, a component used for the X-ray generation becomes obsolete, because a supplier is no longer available. This causes a redesign of the X-ray generation subsystem. Therefore, the X-ray generation subsystem is used as the design perspective. Of course, effects on other subsystems as well as the key drivers are addressed during this design process (see interview 11, Appendix A).

A final example combining the approaches is that a customer might not be satisfied with the image quality of an interventional X-ray machine. Of course it is possible to look at the subsystem that is mainly responsible for the image quality; the image processing subsystem. However it is also possible to create a FunKey diagram [Bonnema, 2008] of the system that will help to identify the functions that are affecting the key driver image quality. This helps to create a system-wide overview with respect to the key driver image quality and consider the system from that perspective. In this manner, it is possible to decide to focus improvement efforts on the display rather than the image processing software. This shows that these perspectives are not mutually exclusive and that a conceptual system design process can and probably should use these concurrently. The concurrent or at least rapidly subsequent consideration of different perspectives is described by [Muller, 2004] as viewpoint hopping. In this way, the big picture is maintained which ensures that all concerns are addressed.

3.1.3 Conclusions

In conclusion, a method supporting conceptual system design should support a system approach, as well as a key driver approach and a subsystem approach. Furthermore, it should take into account the various types of design processes that occur in industry and the specific knowledge sources that are associated with those design processes.

3.2 Behavioral Analysis in Conceptual System Design

This section reflects on the role and place of behavioral analysis in conceptual system design. The goal is to define in what manner the conceptual design of systems is impacted by behavioral analyses.

Nowadays, systems tend to only increase in complexity, as customers and end-users request increased functionality and performance. The resulting system is often an expansion of an already existing system. This expanded system has an increased number of elements that interact with one other with respect to its previous version. These interactions need to be understood to ensure that the system does what it is supposed to do under every circumstance. [Walden *et al.*, 2015] refer to this as robustness and define robustness as the ability to adapt to projected future needs and interoperating systems. In addition to this, [Boardman and Sauser, 2008] introduce the concept of resiliency to indicate whether a system can return to its normal operating modus following upheavals in the context of system.

Understanding interactions requires approaches that focus on thinking about dynamic and operational aspects of the system. An example of this are the dynamic and operational thinking tracks proposed in [Bonnema, 2012]. To concretize these lines of thinking, various

design questions can be envisaged using the WHWWW question generator [Muller, 2004]. Examples of these kind of questions are:

- How is [key driver x] affected by elements across the system?
- If we change a property of [view x] in [subsystem x], how does it affect the [other views] in [subsystem x], or how does it affect the [key drivers] of the system?
- What is the value for the customer/user if [key driver x] is improved?

These question are general and answering them does not require behavioral analysis per se. Questions that specifically focus on the system's behavior could be:

- How does [stakeholder x] initiate [function x] when the context changes to [factor y]?
- If the system is performing [function x], is [function y] also available?
- What is the value of [parameter x], influencing [key driver x], during operation of the system?
- How do changes in contextual [factor x] and [factor y] influence the [key driver x] or the execution of [function y]?

Design discussions that debate these questions require support. For simple problems, a drawing might suffice as a mental model to support the discussion. Once systems become increasingly complex and multiple factors are relevant, context diagrams can give an overview of relevant interactions [Boardman and Sauser, 2008]. Also, depicting the system across various use cases and scenarios helps tremendously in this regard. However, these static analysis techniques have certain limits. These limits are reached if for example the number of use cases becomes too large, the system has many configurations that influence its behavior or if the analysis required is too complex for manual analysis. This is where automated models and simulations play a role.

In conclusion, behavioral analysis can be done in many ways and is an important aspect of conceptual system design. In the decision making process it mainly supports problem understanding and analysis. These are the initial two steps in the CAFCR Decision making approach [Muller, 2004] shown in Figure 2.6. Without behavioral analysis, it would not be possible to develop systems that behave in predictable ways and are able to cope with a changing system context.

3.3 System Architecture in Conceptual System Design

The previous sections have shed some more light on several key points of the conceptual system design process. This leads to the question what the actual representation of the system architecture should be during and as a result of the conceptual system design phase. The aim of this section is to define which views are considered to be relevant for the type of application and system that is considered in this research.

The review of existing system architecture frameworks in section 2.2.2 outlined the relation between concerns, views and viewpoints and discusses which views could be relevant. [ISO/IEC/IEEE, 2011] does not prescribe any specific views, but does specify that a system architecture description is composed of one or more architecture views that each govern a specific viewpoint. Note that there is no single "system architecture view" as separate views together describe the system architecture.

3.3.1 Views and Viewpoints Supporting Conceptual System Design

Section 3.1.2 identified three types of perspectives in the conceptual system design process in industry; (i) the overall system perspective, (ii) the subsystem perspectives and (iii) the key driver perspectives. However, what is the relation of the term perspective with respect to the architecture description concepts in [ISO/IEC/IEEE, 2011]?

To clarify this, the definitions for views, viewpoint and concerns from [ISO/IEC/IEEE, 2011] are considered.

Architecture view; work product expressing the architecture of a system from the perspective of specific system concerns

Architecture viewpoint; work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns

Concern; (system) interest in a system relevant to one or more of its stakeholders including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences

Based on these definitions, it is possible to conclude that an architecture view is the result of applying a viewpoint to a specific system concern. The aforementioned perspectives can be aligned with concerns, as they represent a specific interest in the system. For the key driver perspective, this is very clear, as key drivers represent the essence of the objectives of the customers, their concerns. The overall system perspective can be considered as the generic concern for a system. Finally, it seems unnatural to consider the subsystem perspective as a concern directly. However, the subsystem perspective represents the organizational interests in the system, as the subsystems most often dictate how an organization is shaped and how it approaches the design process. These organizational interests can be interpreted as a concern, according to the definition. Therefore, the subsystem perspective can be considered as a concern as well.

A logical question is whether these three types of concerns are exhaustive. By defining the key drivers for the system, a system architect already has made a selection of which concerns are most critical and relevant to the design process at hand [Muller, 2004]. In order to do this, the full system must be considered in its context first, which is what the system concern represents. This analysis of the context includes identification of outside elements that influence the system as well as the identification of stakeholders and their needs. [Topper and Horner, 2013] refer to this as a domain model. In the end, the overall system concern will address the system in its context and support identification of all other concerns. Finally, the addition of the concerns with a subsystem perspective stems from the fact that most companies are organized around subsystems. It could be argued that using a concern based on a subsystem could hamper the conceptual system design process. It might encourage designers to focus on the solution domain and thus consider the problem domain insufficiently [Bonnema *et al.*, 2015, Morris *et al.*, 2015]. However, using the subsystem as a concern does not dictate the particular viewpoints that are used to regard that concern and contextual or functional viewpoints that allow for more abstraction can

still be used to construct views for a subsystem concern. To validate this reasoning, this issue needs to be explored in more depth, which will be done in Chapter 5 and Chapter 6.

DoDAF [U.S. Department of Defense, 2010] propagates the use of a "Fit-For-Purpose" architecture. This means that an architectural description should be consistent with the objectives of a specific project. In this research, that specific project is the conceptual system design phase of a complex system, which requires representation of the three aforementioned concerns. DoDAF acknowledges the possibility to construct "Fit-For-Purpose" views. This means that custom architecture descriptions with custom views are recognized to have value within the DoDAF framework, even though the DoDAF framework specifies an extensive number of viewpoints that can be used to construct views. For example, DoDAF specifies views focusing on capability, data and information, projects, services and standards. The description of these views includes the description of corresponding viewpoints.

To determine the essential views in communication of system behavior, various literature sources were considered. The Y-chart modeling approach [Kienhuis *et al.*, 1997] is used in the Boderc design methodology [Heemels and Muller, 2006] to model and simulate in the conceptual stages of systems engineering. The Y-chart methodology uses an application (functional) view and a platform (physical) view and combines those with a mapping. A graphical description of this approach is shown in Figure 3.2. However, before simulation is possible, both of the views need to be quantified as well. This is akin to the triad of a functional view, a physical view and a quantification view as recommended in [Bonnema, 2014]. This triad of views supports multidisciplinary communication on complex systems.

Furthermore, [Topper and Horner, 2013] discuss modeling in the conceptual system design stage and identify relevant components of such a model, enabling subsequent simulation. They state that a functional view and structural (physical) view are key in conceptual modeling, as well as a description of use cases and a domain model. However, there is no explicit mention of a quantification view. DoDAF [U.S. Department of Defense, 2010]

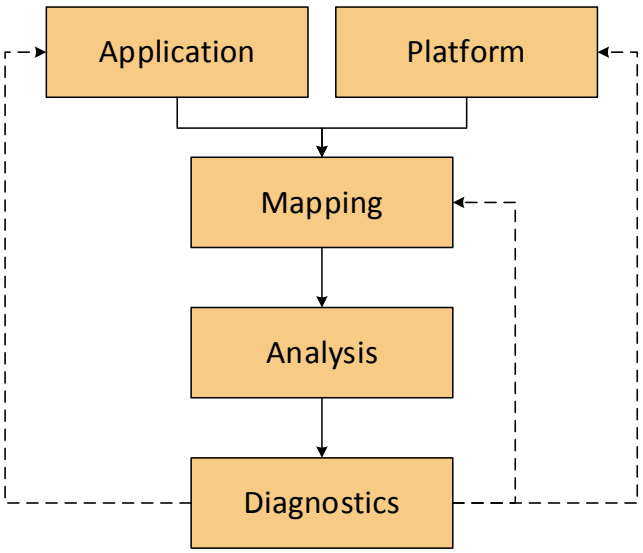


Figure 3.2 – The Y-Chart paradigm maps an application view to a platform view. The results of this process can be analyzed. Subsequently, either application, platform or mapping can be adjusted. Graphic reproduced from [Basten et al., 2010]

specifies a fairly large number of possible viewpoints. Relevant viewpoints are amongst others the all viewpoint (the AV-1 product as problem definition), the operational viewpoint (OV-1, CONOPS diagram) and the systems viewpoint (SV-4 Systems Functionality Description).

The operational viewpoint in DoDAF and in [Topper and Horner, 2013] (through use cases) provides the possibility to explore various scenarios in which the system can operate. This helps greatly in assessing risks and uncertainty for the system. Therefore, it is clear that an operational view is also essential to gain insight in system behavior during the conceptual system design stage. The operational view helps to specify uncertainty due to changes in the environment and helps to consider a time aspect for the system. However, there is also internal uncertainty in the system, for example the final value of a parameter. These kinds of uncertainty are better represented in the quantification view.

In essence, the four viewpoints (functional, physical, quantification, operational) that have surfaced in this discussion are essential to construct useful views in conceptual system design. This is because the combination of these views allows both multidisciplinary communication and the connection to simulations. Other viewpoints can be relevant as well on a case by case basis, such as making the context more explicit in a domain model.

3.3.2 Addressing the Overall System Concern

This subsection discusses which viewpoints can be used to create views for the overall system concern.

A view that describes the overall system from the context viewpoint is a context diagram. An example of such a diagram is a systemigram [Boardman and Sauser, 2008], which is shown for an iXR Allura System (see also section 1.3.1) in Figure 3.3. A context diagram can include factors such as the wider system that contains the system of interest, contextual influences and how they impact key drivers, functions and stakeholders. [Boardman and Sauser, 2008] specify a number of design rules for systemigrams. For example, Boardman and Sauser state that systemigrams must fit on a single page and there should be no crossover of links, improving clarity. Boardman and Sauser also state that this last constraint lends itself to systemic design as it reduces the number of interfaces. Finally, they state that color can be used to draw attention in a consistent way

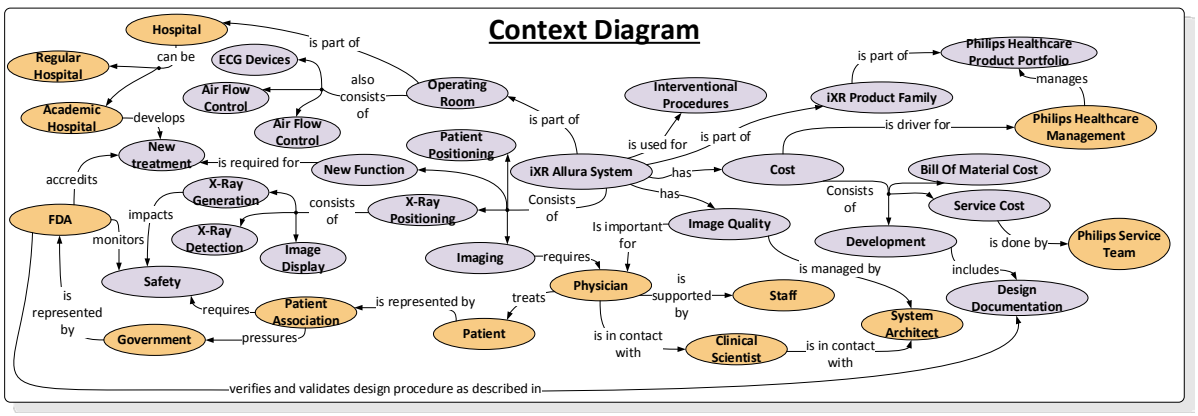


Figure 3.3 – Example context diagram (or systemigram [Boardman and Sauser, 2008])

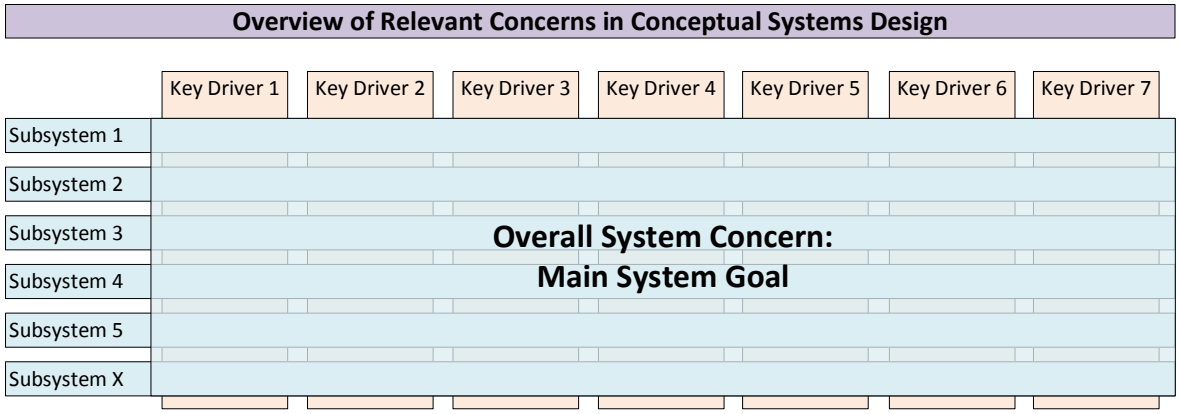


Figure 3.4 – Concerns View in Conceptual System Design (based on [Bonnema, 2008])

to subfamilies of concepts and transformations. Groupings are also allowed. In the example in Figure 3.3 a different color is used to indicate stakeholders.

At the system level, it is possible to represent an operational view with for example use cases. These can be derived from the main purpose or goal of the system. Several high level use cases can be devised to characterize how the system should operate. An example method that can be used to define these use cases is storytelling [Muller, 2004]. Muller states that the story can be represented by text supported with one or several figures. The story should be specific and focus on regular use scenarios. However, the scenario should be challenging as well.

A view describing all concerns should also describe key driver and subsystem concerns. Key drivers can be inferred from the context diagram and discussions with stakeholders. The context diagram and the operational view are helpful tools to identify the functions. An approach that uses key drivers and functions to define the system architecture and ultimately divide the system into subsystems is FunKey architecting [Bonnema, 2008]. While a FunKey diagram could be used as a view to visualize the concerns, it places functions in a central role instead of subsystems

Therefore, this research uses a visualization that differs slightly from a FunKey diagram and can be seen in Figure 3.4. An important aspect that is retained from the FunKey diagram visualization is the fact that key drivers are visually crosscutting all subsystems. The visualization includes the definition of the main system goal, as well as the subsystems and key drivers.

Summarizing, the views describing the overall system focus on the context and the key drivers and functions that can be derived from this context. These functions can be organized into subsystems. Of the essential four viewpoints discussed in the section 3.3.1, the physical viewpoint is missing. However, the physical viewpoint is less relevant at the system level as it is when representing key driver and subsystem concerns.

3.3.3 Addressing Subsystem and Key Driver Concerns

The four essential viewpoints that were identified can be used to create views for the subsystem and key driver concerns as well. Table 3.2 summarizes several of the possible models that can be used according to these viewpoints.

Table 3.2 – Overview of essential viewpoints and diagrams usable to construct views

Viewpoint	Models provided by viewpoint to construct views
Functional	Functional Flow [Borches, 2010, p91] Functional Hierarchy/Decomposition [Blanchard and Fabrycky, 2010, p96] Functional Value Analysis (see Interview 1, Appendix A) UML/SysML Activity and State Diagrams [Topper and Horner, 2013]
Physical	Block diagram [Borches, 2010, p92] 3D CAD Model [Hooman *et al.*, 2012] SysML block definition diagrams [Topper and Horner, 2013] Bill of Materials (see Interview 1, Appendix A)
Quantification	Budgets [Bonnema, 2008, p64, Freriks *et al.*, 2006] Quantifying Key Parameters [Borches, 2010, p92]
Operational	Scenarios, event flows and other behavioral models, see Table 2.5 Story telling [Muller, 2004]

Other viewpoints than the four essential ones might be relevant as well on a case-by-case basis. However, this will have to be validated at a later stage through exploration in case studies. Therefore other viewpoints and resulting views are excluded in Table 3.2, as well as in Figure 3.5, which is discussed in the next subsection.

3.3.4 System Architecture Views in the Conceptual System Design Stage

This subsection concludes the investigation into which views are required to describe a system architecture in the conceptual system design phase. An overview of these views is given in Figure 3.5. This figure shows that the overall system concern can be described with four different views. One of these views is a concerns view as presented in Figure 3.4. The concerns described in this view can be described using views that represent the four essential viewpoints.

3.4 Conclusions

This chapter discussed the conceptual system design process by using observations and interviews in industry. This resulted in identification of four types of conceptual system design processes, which either concern a new system, a system redesign, a subsystem rede-

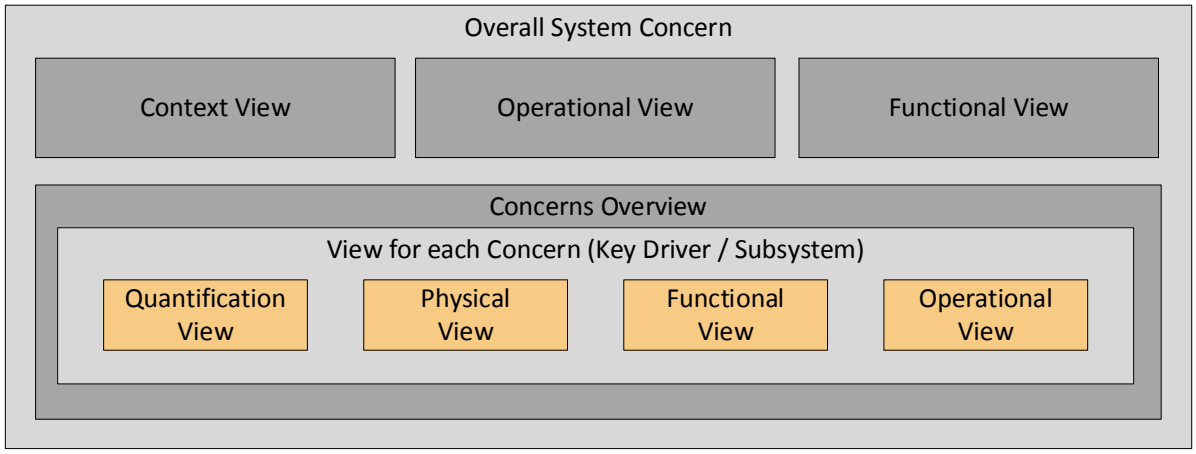
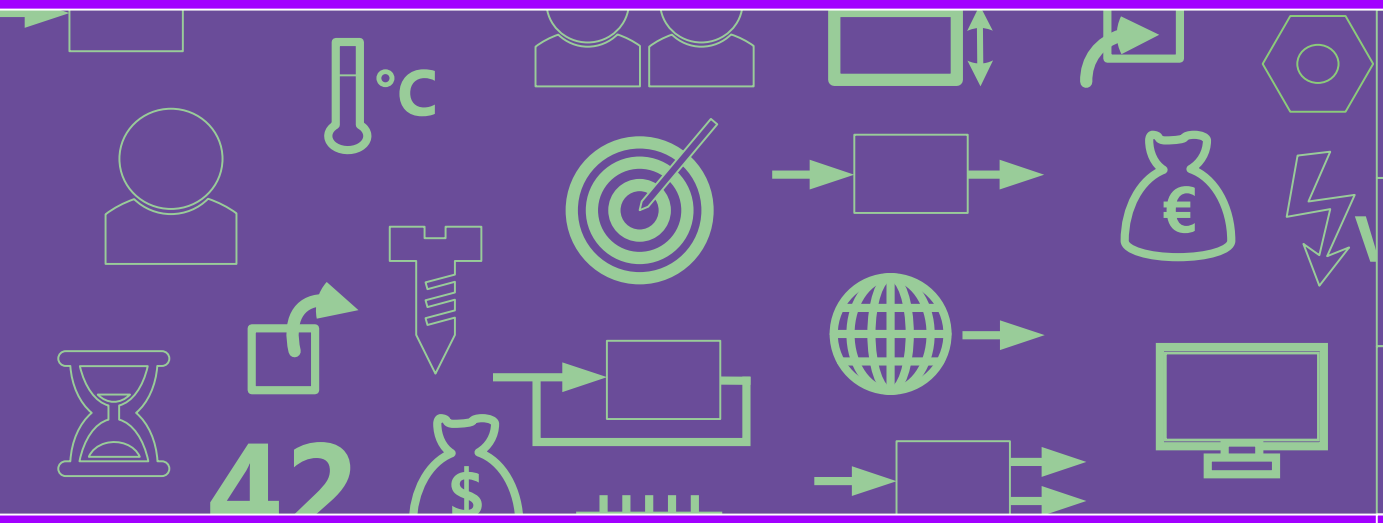


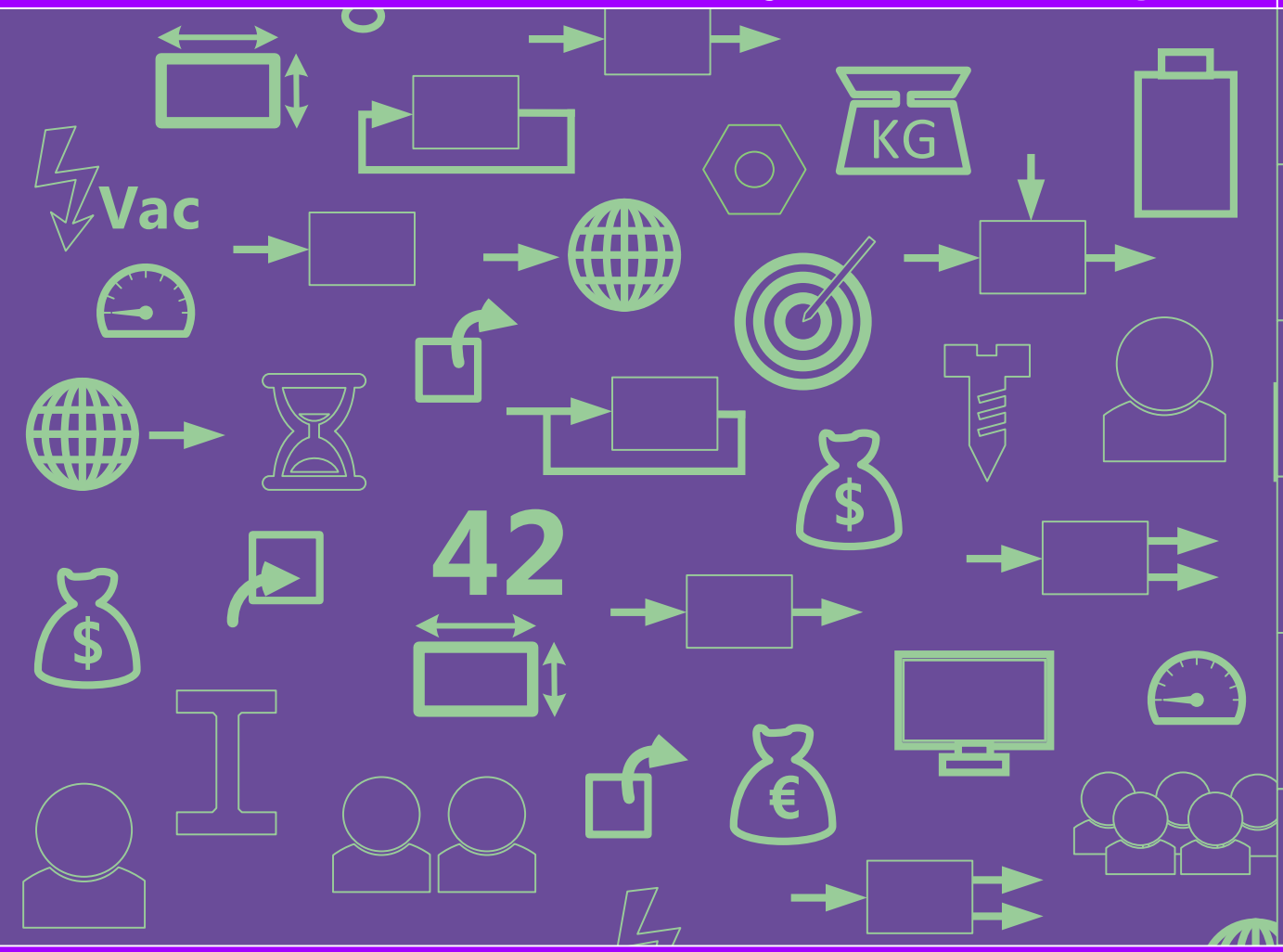
Figure 3.5 – Views describing a system architecture in the conceptual system design stage

sign or a component update. Additionally, various design questions that need to be answered during conceptual system design were discussed. These questions exemplify the role of behavior analysis.

Furthermore, three perspectives to consider the conceptual system design process were identified, either reasoning from the overall system perspective, a key driver perspective or a subsystem perspective. The identified perspectives led to the conclusion that these are the concerns that need to be represented with architectural views to describe the architecture during conceptual system design. For each of the identified concerns, the architectural views that can be constructed based on different viewpoints were discussed. The essential viewpoints to construct these views are the functional, physical, quantification and operational viewpoint. These viewpoints serve both as a representation of the system behavior and as an excellent foundation for multidisciplinary communication.



Communicating Simulations in Conceptual Systems Design



4

Communicating Simulations in Conceptual System Design

This chapter is partly based on research that has been published in two conference papers. These are [Haveman and Bonnema, 2015a] and [Haveman and Bonnema, 2015b].

This chapter provides a new view on simulation processes in conceptual design. The chapter starts with an in-depth literature review of how simulations are currently used in conceptual system design in section 4.1. This review is used to perform an analysis of communication support for simulations. Based on this analysis, a description of a generic simulation process in conceptual system design is given. Section 4.2 defines several challenges that need to be addressed in order to be able to communicate simulations effectively in the conceptual system design stages. This is summarized in a compact framework that gives guidance on how to approach simulation studies if these challenges are to be mitigated. In section 0, the key elements that constitute models and simulations in the conceptual system design stage are identified, especially those related to communication. These concepts are brought together in a reference model that extends the [ISO/IEC/IEEE, 2011] conceptual model. This model can act as a frame of reference to support communication of modeling and simulation studies.

4.1 Simulations in Conceptual Systems Design

The objective of this section is to consider simulation studies in more detail. This is done by reviewing simulation methodologies with respect to conceptual system design. Based on this review, a generic simulation process in the conceptual system design stage will be defined.

4.1.1 Simulation Methodologies

As simulation and modeling have always been important concepts inside and outside of systems engineering, there is an extensive body of knowledge on how to support simulation studies. Table 4.1 summarizes a literature review of several simulation methodologies based on: (i) their purpose, (ii) the means with which they achieve this purpose, (iii) the main outcomes of using the methodology and (iv) the tools, languages and methods used. The selection of methodologies in Table 4.1 consists of:

- Domain independent methodologies

[Birta and Arbez, 2013, Boardman and Sauser, 2008, Browning *et al.*, 2006, Law, 2014]

- Systems engineering specific methodologies

[Bjorkman *et al.*, 2013, NASA, 2008, Ryan *et al.*, 2014, Wang and Dagli, 2011]

- Methodologies with a focus on the conceptual system design stage

[Basten *et al.*, 2013, Canedo and Richter, 2014, Topper and Horner, 2013, Yaroker *et al.*, 2013]

Table 4.1 – Overview of Simulation Methodologies

Methodology Name (and Reference)		**Purpose**
Means	**Main Outcomes**	**Tools & language**
Simulation Modeling and Analysis [Law, 2014]		Give overview of simulation process, avoid heuristic model building, programming and a single simulation
A generic simulation study process description with emphasis on validation and verification	*- Design and analysis support* *- Determining requirements*	*Written assumptions document* *No specific tools*
Modeling and Simulation: Exploring Dynamic System Behavior [Birta and Arbez, 2013]		Developing a meaningful representation of the System Under Investigation
Activity-Based Conceptual Modeling	*- Leveraging of behavioral data* *- Capture of relevant details, avoid superfluous features*	*ABCmod conceptual model* *Three-Phase simulation model*
Systems Thinking: Coping with 21st Century Problems [Boardman and Sauser, 2008]		Organize thoughts and actions relative to the system of interest
Framework for systems thinking, considering product, process and enterprise as a whole	*- Models reflect system and serve as discussion tool* *- Insight in relevance of different views*	*Systemigrams* *Soft Systems Methodology*
Key concepts in modeling product development processes [Browning *et al.*, 2006]		Integrating the disparate models in use across an organization
Using activities and deliverables as key concepts in a generalized product development framework	*- Structure and reuse knowledge* *- Improving organizational, tool and product integration*	*PD process model*
NASA STD 7009 [NASA, 2008]		Offer critical decision support
Standardize / Certify and document simulation procedure	*- Assurance that the credibility of models and simulations meet project requirements*	*COTS tools* *Delphi Method [Ahn et al., 2014]*
MBSE to Improve Test and Evaluation [Bjorkman *et al.*, 2013]		Systematically reducing uncertainty
Coupling of simulation and test results	*- Uncertainty predictions are easy to obtain and visualize*	*SysML/UML* *Monte Carlo*
Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis [Ryan *et al.*, 2014]		Representing sophisticated design options
Extending parameterized trade studies with variability modeling	*- Current configuration of design decision;* *- Increased communication* *- Traceability and potential for SE reuse*	*SysML* *Matlab* *Excel*
Executable system architecting using SysML in conjunction with CPN [Wang and Dagli, 2011]		Static and dynamic system analysis and formal verification
Conversion of SysML-based specifications into colored Petri nets	*- Visualize a proposed system* *- Analyze the problem domain* *- Specify architecture for solution domain.*	*SysML* *Colored Petri Nets (CPN)*
Architectural Design Space Exploration of Cyber-Physical Systems using the Functional Modeling Compiler [Canedo and Richter, 2014]		Evaluate the system-level impact of domain-specific design decisions
Functional modeling to perform architectural DSE using multi-disciplinary simulations	*- Detailed multi-domain design space exploration*	*Functional Modeling Compiler (FMC)* *AMESIM / Modelica*
An OPM Conceptual Model-Based Executable Simulation Environment [Yaroker *et al.*, 2013]		Find mismatches between design and requirements considering dynamic aspects
Object Process Methodology Model-Based Simulation	*- Improved behavioral analysis* *- Degraded structural analysis*	*Object Process Diagram* *OPCAT*
Model-Driven Design-Space Exploration for Software-Intensive Embedded Systems [Basten *et al.*, 2013]		Systematic evaluation of design choices early in the development
Intermediate representation allowing connection of tools and techniques	*- Integrate languages and tools in a unifying framework*	*DSE Intermediate Representation* *CPN / Uppaal / SDF3*
MBSE in Support of Complex Systems Development [Topper and Horner, 2013]		Understanding of critical components , interfaces and processes
Conceptual modeling using a light weight, agile approach (ICONIX)	*- Facilitate communication & collaboration* *- Reuse components & results, improve traceability, information management*	*ICONIX* *UML / SysML*

Many methodologies focus solely on later design stages. For example [Ryan *et al.*, 2014] even mention:

"once the requirements are understood, the trade space is defined and potential architectures have been identified, simulation models can be designed"

Simulations are certainly very important in these later stages of design. However, all the activities that are mentioned in this quote can be supported with simulation models as well. [Canedo and Richter, 2014] state that determining the impact of new design alternatives is not supported well by state-of-the-art design tools. In addition to this, [Yaroker *et al.*, 2013] state that conceptual system design is a crucial system lifecycle stage, but systematic methods for conceptual system design evaluation are not well developed.

4.1.2 Role of Simulations in Conceptual System Design

Everyone that has performed an analysis using simulations will recognize that when the simulation is "finished", the modeler knows "everything" about that aspect of the system. Of course, this is a good thing. However, which part of what has been learned needs to be shared with other stakeholders? As stated in the introduction, sharing insights and results during the simulation process is important [Morse *et al.*, 2010, Topper and Horner, 2013]. This requires a different approach than just creating a model and simulating it. This is reflected in the following statement by [Birta and Arbez, 2013]:

"It is never meaningful to undertake a study whose goal is simply 'to develop a model of -' "

With this statement, they emphasize that a goal should be defined before starting the modeling and simulation activity. In the conceptual stages of SE this goal is often to define the problem. The models required for this approach should be only as complex as necessary to meet the objectives of the simulation study [Robinson, 2008a]. This requires a different approach than for example searching for an optimal configuration of the system in a trade study. [Boardman and Sauser, 2008] emphasize the following:

"We believe that simulating the product (or service) has had a fair crack of the whip. It is time for visibility into the black box to convey confidence that what will emerge will be what people really need. We call this competence demonstration or enterprise realization assurance, as opposed to technology demonstration"

With this quote, [Boardman and Sauser, 2008] state that the capability to create simulations of systems is omnipresent. However, the capability to share the knowledge that is contained within this simulation is lacking. This notion is especially important in conceptual systems engineering, to connect stakeholders and illustrate the impact of the system on those stakeholders. The simulation methodologies presented in Table 4.1 do not explicitly mention this concept. [Boardman and Sauser, 2008] themselves indicate the need for this kind of simulation, and offer systems thinking as a generic tool in this regard, but have not worked towards an actual implementation.

A lot of emphasis is put on several communicative aspects of a simulation study in what is considered the "Bible" on simulation [Law, 2014]. One of these is maintaining a written assumptions documents and another is to interact with the manager on a regular basis.

Also, having a structured walkthrough of the assumptions document with all relevant domain experts is highly recommended. However, no guidance is offered as to how modelers should select information or structure its presentation.

4.1.3 The Simulation Process in Conceptual System Design

The previous sections showed that simulations are a great tool to gain insight in the behavior of a system. However, communication of simulations is complicated. First, there is no explicit knowledge base on how to actually perform this communication. Most simulation frameworks focus on the technical side of the simulation (setting up experiments, doing statistical analysis, etc.). They often end with the note "and communicate your results". Second, not only the end-result should be communicated, but also the process of defining, designing and analyzing the underlying model. Therefore, the focus must be on both the process and end-result when communicating simulations.

To do so, this section presents a simulation process that explicitly mentions communication and the relation between problem and solution domain. This is done by adapting work of [Law, 2014]. [Law, 2014] was selected because this is one of the most well-known descriptions of simulation studies and is one of the few that emphasize communication. The resulting simulation process with an explicit distinction between the problem and solution domain has been described in [Haveman and Bonnema, 2015a]. This process is visualized in Figure 4.1. It consists of six steps of which the sixth step is the validation and communication of results. However, this activity takes place during the whole process, indicated by the fact that the block moves up on the right side next to the other blocks. The distinction between problem and solution domain is indicated by the blocks on the left.

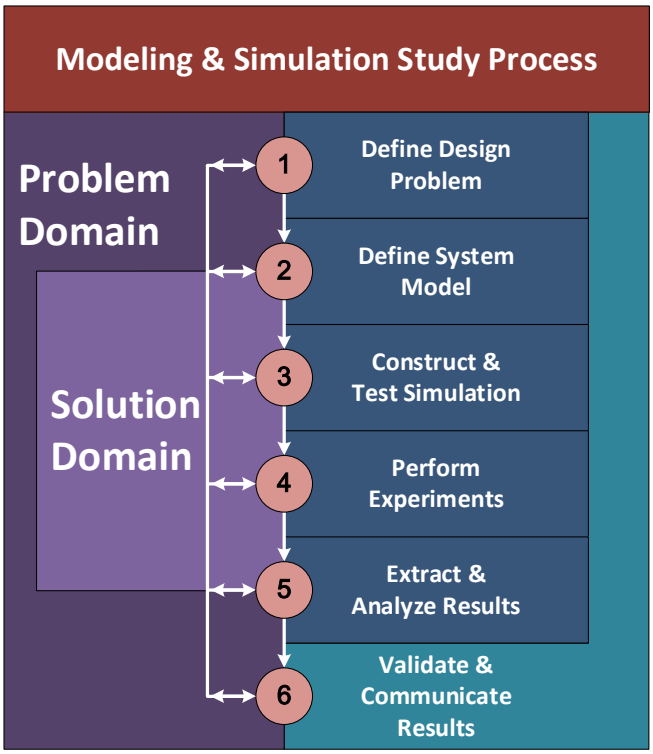


Figure 4.1 – A simulation study process applicable to conceptual systems design [Haveman and Bonnema, 2015a], based on [Law, 2014]

This means that step 1 & 6 are fully in the problem domain and 2 & 5 partly. The other steps are in the solution domain. The arrows indicate that the process is iterative and can return to any previous step. Section 4.3.2 shortly discusses these steps, but [Law, 2014] and [Haveman and Bonnema, 2015a] describe the individual steps in further detail.

The process described in Figure 4.1 can be used as a reference when performing simulations in conceptual system design. This process is embedded in the method presented in Chapter 6, but can be used for stand-alone applications as well. The main reason for creating and describing this process is to support a transition between generic simulation study process descriptions and one that focuses on the conceptual system design stage and emphasizes the importance of communication.

4.2 Key Challenges

The previous section outlined that it is not straightforward to utilize and communicate simulations in conceptual system design. Apparently, several barriers exist to do so. This section will explore these barriers, by first identifying them and discussing how these challenges can be mitigated.

4.2.1 Identification of challenges

To identify the challenges that inhibit usage of simulations in conceptual system design, the conceptual system design stages will be compared with detailed design stages where their usage is more common.

First of all, the conceptual system design stage is multidisciplinary in nature. This means that the input stems from multiple disciplines, but also that the output of the step has to be communicated to multiple disciplines as well. This happens for example during design reviews, in which key personnel from various divisions is involved. In interviews in industry it has been indicated that these reviews tend to be complicated and tedious, caused by a lack of insight that stakeholders have in the system and by the many different viewpoints that have to be accounted for (see Appendix A, interview 4 and 5). Each discipline has its own jargon and formalisms and when simulations are used in detailed design, they make explicit use of these formalisms. However, in conceptual system design the simulation and the model behind it need to resonate with multiple disciplines and cannot be communicated using a single formalism that stems from one of these disciplines.

Second, another distinct difference is that in conceptual system design, design space exploration has a much more explorative nature [Theelen and Hooman, 2015], whereas the more detailed stages of design are more of a problem solving nature. During this divergent design space exploration, there is a large focus on creative thinking, to define and try out new architectures. Design space exploration in detailed design can be termed convergent and aims to find a system design that solves the problem, i.e. meets the requirements. This type of design space is often more constrained as the goal is optimization. Such an approach is widely supported [Gries, 2003].

Third, there is a large difference in the degree of uncertainty between early and detailed design. In both cases, it is necessary to estimate an input to determine performance. At the same time, in early design parts of the system might still be completely unknown. For

Table 4.2 – Overview of challenges in communication of simulations in conceptual system design

Accommodate multiple disciplines
In the conceptual stage of systems engineering, a broad audience is involved. Where possible jargon should be avoided and insight in every discipline involved should be offered
Support divergent design space exploration
In the early stages of systems engineering the goal is not to optimize a system, but to explore and consider multiple options. In this stage, thinking out-of-the-box is important. This in turn means that there should be a greater emphasis on the problem domain.
Dealing with uncertainty
Uncertainty can come in various forms and shapes. This can be either uncertainty in a parameter, but more importantly uncertainty in the systems functionality or even uncertainty in the systems context.
Lack of formality
Conceptual systems engineering and multidisciplinary communication is informal by nature. However, simulations usually require a significant measure of formality.

example, during conceptual system design of a car, the propulsion method might not be decided yet. This complicates insight in for example driving behavior. In detailed design, using the same car example, the early decision might have been to use an electric drivetrain, but the size of the battery packs is still unknown. In this case, the general behavior of battery packs is already known, so a simulation model can be established much easier.

A fourth and final difference between conceptual system design and detailed design is that conceptual system design is much more informal in nature. While initiatives such as the MBCD WG [INCOSE, 2013] aim to increase the formality in conceptual system design, the fact remains that it is fairly informal. Designs are loosely described and drawings or sketches are used to clarify concepts. This approach is logical from a conceptual system design perspective, as this informality can be necessary to overcome discipline boundaries or cope with uncertainty. However, simulations require a model that is defined in such a way that it is executable, which requires formality.

The four differences that have been discussed give a clear indication as to why simulations are difficult to employ. The identified challenges are summarized in Table 4.2. In order to enable a successful application of simulations in conceptual system design, these challenges need to be addressed. The following subsections discuss how this can be approached.

4.2.2 Accommodate Multiple Disciplines

In order to accommodate multiple disciplines with a simulation during conceptual system design, a paradigm shift is needed. Traditionally, simulations are used to support analysis within single disciplines. Physics based simulations have succeeded in bridging gaps to other engineering disciplines, but not to the "non-technical" disciplines. While effort has been put into aligning the concepts and languages used by various disciplines to create a universally understood language, this has so far been unsuccessful [Torry-Smith *et al.*, 2011]. Furthermore, in conceptual system design, many stakeholders are loosely involved in the design

project. Especially when using a user centered design philosophy that for example uses ad-hoc focus groups. Therefore, the process will involve multiple stakeholders who lack time and resources to acquaint themselves with modeling languages such as SysML.

In addition to this, [Rozanski and Woods, 2012] state that in software engineering it is impossible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable by, and of value to, its stakeholders. As one model seems to be a holy grail [Torry-Smith *et al.*, 2011], multiple views have to be used that cover and relate to a wide and essential range of stakeholder viewpoints. The A3AO method as described by [Borches, 2010] uses this approach and has proven to be successful as a multidisciplinary communication tool in industry. When the A3AO method is compared to the views presented in section 3.3.4, the only view lacking to be able to communicate system behavior is the operational view. However, [Muller, 2014] discusses a subsea A3 Architecture Overview showing various workflows. This is done in a "comic book" style, meaning that various subsequent images show the system's state over time. In [Singh and Muller, 2013], a dynamic A3 architecture is used which allows various use cases of a lube oil system to be viewed. Their digital implementation presents an overview with hyperlinks to more detailed A3's showing use cases such as "start-up", "running" and "shutdown". These two works give a good indication of how an operational view can be employed in an A3, either in the classical paper format [Muller, 2014] or digitally, in [Singh and Muller, 2013]. However, it can also be imagined that more interactivity is required to accommodate multidisciplinary views. Additional interactivity allows users to define operational scenarios themselves and directly see the resulting system behavior.

An approach by [Ivanovic and America, 2010] involves disciplines by quantifying the impact of design choices on both customer value and other stakeholders value. Here it is observed that a mere technological assessment may overlook the main customer needs. A clear overview of the key drivers of a system is therefore necessary, as they help to focus the development [Muller, 2004]. Architecture-driven quality requirements prioritization [Koziolek, 2012] is described as a technique to automatically analyze trade-offs between different quality requirements.

In essence, to accommodate multiple disciplines, the use of multiple views is a feasible solution which has proven itself in the A3AO method [Borches, 2010]. Furthermore, the impact of design choices should be exemplified across the concerns of multiple stakeholders.

4.2.3 Support Divergent Design Space Exploration

Simulations used in systems engineering are often focused at optimization of an architecture or system design, given a set of requirements. For example, multi criteria decision making defines all criteria upfront and then seeks an optimum [Grogan *et al.*, 2015]. This optimum can be found by changing and experimenting with the input parameters for a model. In some cases simulations are used to identify emergent system behavior [Baldwin *et al.*, 2015]. The out-of-the-box nature of divergence makes it is logical that this process mismatches with simulations, as all possible paths need to be programmed upfront. Therefore, there is limited support in simulation techniques to support divergent design space exploration. To support this divergence it is possible to consider approaches that are

currently used in conceptual system design. For example, FunKey architecting [Bonnema, 2008], TRIZ [Altshuller, 1997] or key driver maps [Haveman and Bonnema, 2013, Heemels *et al.*, 2006a]. However, these techniques are not simulation based. To answer the question of how to address this issue when using simulations, several measures have been identified.

First, it must be avoided to employ simulations purely to find an optimum. Therefore, the goal of the simulation study must be clear to stakeholders [Birta and Arbez, 2013]. In the conceptual stages, the goal should be to clearly define the generic problem and not to find an optimum solution for the specific problem at hand. When the solution domain is entered, it should be done with care and at first with an explorative mindset (see interview 9 and 10, Appendix A).

Second, using functional descriptions and analysis is preferable, since they are solution independent [Canedo and Richter, 2014]. So, the goal of a simulation model should be to support basic behaviors that aim towards broad understanding.

Third, when parameters are introduced, one should always be able to vary these, because this allows for what-if exploration. When performing experiments, the variation of these parameters should be used to find key concepts that determine the behavior of the system. This includes thinking about the system's dynamic characteristics and feedback loops.

Fourth and finally, when presenting the results, they should be presented in a way that leads discussion towards the problem and not towards the solution. For example, by avoiding presenting a specific system model, but to merely present and discuss the generic issues that dictate the system's behavior.

4.2.4 Dealing with Uncertainty

Inherently, simulation studies in the conceptual stages of systems engineering have a high uncertainty. When defining the design problem, this will surface due to the fact that there is no architecture concept or it is still vague. When defining the system model, it is key to also quantify the context of the system and consider possible risks. To handle and model this uncertainty, both formal and practical approaches are possible [Weck *et al.*, 2007]. A common formal approach is to use fuzzy math to account for ranges of estimations. GuideArch [Esfahani *et al.*, 2013] is a framework that uses these techniques to guide engineers to make the best choices possible under uncertainty. Another approach is to acknowledge imperfections and uncertainties and to build your design process to deal with them effectively. For example, one can use design trees [Noppen *et al.*, 2007] in which feasible solutions can be retraced easily if the chosen solution does not suffice. A practical approach, could be to use scenarios [Weck *et al.*, 2007]. Scenarios can be used to envision various internal and external influences to the system and its response to these influences.

Regarding the quantification of parameters, it can be chosen to represent them with a range instead of a fixed numbers [Bonnema, 2008]. During simulations it is possible to randomly sample from this range, or repeat the experiment a number of times with a fixed value for the parameter [Law, 2014]. To visualize this uncertainty, a representation as presented by [Rajabalinejad and Bonnema, 2014] can be used.

Furthermore, when constructing a simulation, generic behavior can be modelled for the system. If this is done flexibly, a great range of dynamic properties can be explored. In the resulting steps, it is important to think about the impact that various inputs from the system context have on the system's output and vice versa.

Finally, when dealing with uncertainty from the context of a system it is necessary to explore that context first using context diagrams to assess possible influences. This allows modeling of what-if scenarios that are relevant to the system's behavior. An example of this can be found in [Ross, 2006].

4.2.5 Lack of Formality

The final challenge that needs to be addressed is the lack of formality in the conceptual system design stage. The Model Based Conceptual Design (MBCD) paradigm argues that more formality is required, as it can provide consistency and traceability [Knight and Vencel, 2014, Woestenenk, 2014]. Furthermore, [Knight and Vencel, 2014] describe that it is becoming more common for complex capabilities to be developed or "grown" using incremental and iterative development approaches, in which project definition is an adaptive process. This makes a strong argument for more formality as it allows a continued and consistent reiteration between a formalized detailed design and the conceptual system design.

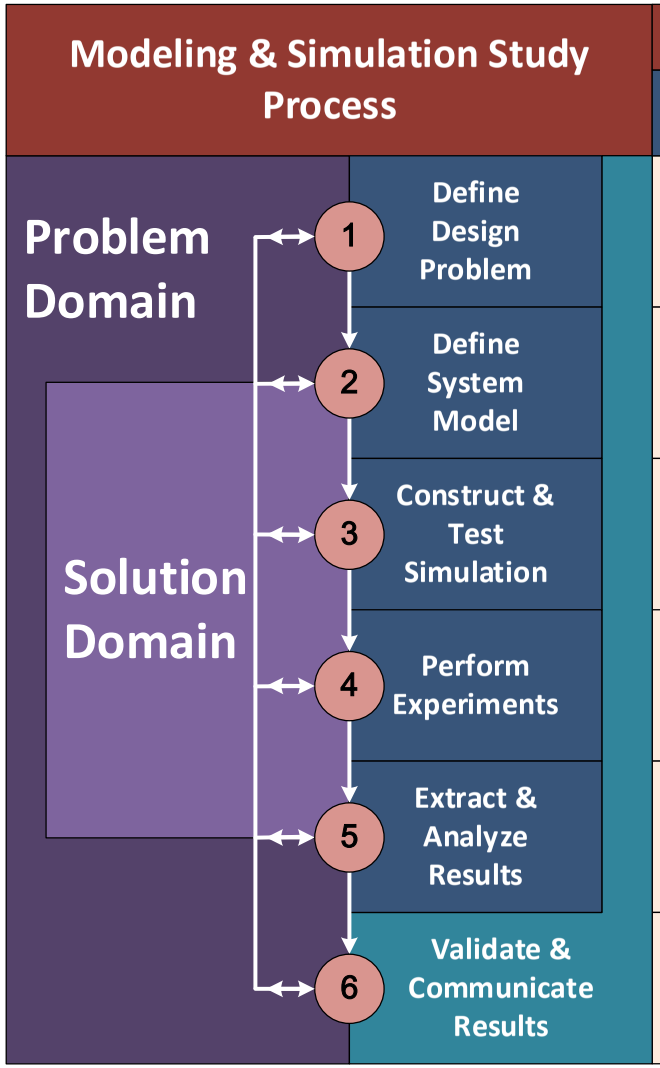
However, as stated before, conceptual system design is informal by nature. Actually, too much formality will constrain and stifle the conceptual system design. [Jakobsson, 2014] presents a Concept Options Development Process for conceptual system design. He argues that MBCD should only be used once a new approach requires structuring and not to support the first stages. However, he does not discuss how to connect the initial steps with later stages. The Y-chart methodology [Kienhuis *et al.*, 1997] lends itself well for modeling and formalizing the system in this stage. It does so by specifying application components and platform components, and mapping them to one another. With a rough quantification of these components, a system model suitable for simulation can be established

The earlier steps that [Jakobsson, 2014] referred to as unsupported by MBCD frame the problem. Problem-framing techniques such as rich pictures are usually "*created with low-tech support, such as whiteboards or pen and paper*" [Valente and Marchetti, 2010]. These drawings help stakeholders to understand the system. [Borches, 2010] refers to these kinds of drawings as visual aids. These visual aids are crucial in capturing design rationale and the problem definition. The Design Framework by [Moneva *et al.*, 2011] allows capturing of these kinds of simple models and can link them to design parameters

In conclusion, formalizing part of conceptual system design is required to enable simulations. However, the creative and informal part of the conceptual system design process resulting in simple models such as drawings need to be accounted for as well.

4.2.6 Framework for Communication of Simulations

This section is concluded by offering a connection to the simulation process that was discussed in section 4.1.3. A compact framework that supports addressing simulations in conceptual system design is presented in Figure 4.2. It does so by opposing the identified.



Modeling & Simulation Study Process	Important viewpoints during the conceptual design stage in Systems Engineering			
	Multidisciplinary Communication	**Problem Focused Approach**	**High Uncertainty**	**Lack of Formality**
Define Design Problem	• Identify stakeholders • Identify stakeholder concerns and key drivers	• The goal is to define the problem (better), not to find an optimum	• No architecture concept present or it can be radically changed still, focus on context	• Fuzzy stakeholders needs are starting point – translate to functions
Define System Model	• Focus on validation of concepts relevant to concerns • Establish credibility by iterating frequently	• Focus on high-level functional and structural elements • Quantifiy using basic parameters	• Structure is undefined or wholly inaccurate, focus on system as a (partial) black box	• Focus on functional modeling
Construct & Test Simulation	• Focus on verification of the systems behavior • Align to existing validation procedures	• Aim towards broad understanding • Allow adjustment of all parameters	• Use ranged parameters • Model generic behavior	• Use a generic simulation model that focuses on a high abstraction level
Perform Experiments	• Focus on validation of the results	• Identify and explore key characteristics that determine behavior	• Sensitivity analysis is key – find influencing factors given system's context	• Focus on behavioral correctness and functional validation
Extract & Analyze Results	• Again, focus on validation of the results	• Aim towards elaborating key characteristics and providing knowledge on generic implications	• Determine influencing factors on (non)-functional performance	• Performance indications can be given based on parameter estimations
Validate & Communicate Results	• Use multiple views • Represent how design decisions impact system concerns	• Lead discussion to how stakeholders wishes are impacted by key characteristics	• Give insight in how design decisions impact system performance • Visualize uncertainty	• Focus on visual aids – its not about the data, its about the concepts behind them

Figure 4.2 – Framework for communication of simulations in conceptual system design

challenges with the simulation process. These challenges are regarded as viewpoints with which the simulation process can be observed. This framework can be employed to improve the use of simulation studies in conceptual system design. However, a more extensive method that utilizes the concepts in this framework is presented in Chapter 6

4.3 Core Concepts

In order to summarize and make sense of all aspects that were discussed so far, this section identifies the core concepts that are relevant for communication of modeling and simulation activities in multidisciplinary systems engineering. The goal is to compile these into a reference model that gives an overview of these concepts and, as the name implies, can be used as a reference in both simulation applications and future research.

To introduce the core concepts of communicating modeling and simulation activities, several topics are addressed. The first is the conceptual system design process in which the simulation study is conducted. The second is the simulation study process itself. The third is the relevant viewpoints and corresponding views that are crucial in this type of activity. Finally, a unified view on the discussed concepts is presented. In this text, the key concepts that will be compiled into a reference model are bolded.

4.3.1 Development Process

Various **design methods** can be employed to develop systems. These can either be document driven or model driven. When considering the position of **modeling and simulation studies** in these processes, they can be either considered loose entities or tightly coupled to the design method, for example by integrating simulations and models with test and evaluation activities [Bjorkman *et al.*, 2013]. A study will also be executed at a certain **abstraction level**. As the main concern is the conceptual stage of systems engineering, high levels of abstraction are most relevant. Within one abstraction level, a simulation study can be iterated, meaning that the same abstraction level is once again considered. Executing the same study at a lower abstraction level is considered recurrence [Bonnema, 2008].

4.3.2 Modeling and Simulation Study

While the development process gives the context for a simulation study, the study itself is a process as well and was described in Figure 4.1. This section reviews the steps in this process one-by-one and relevant concepts are identified.

Problem Definition

Due to the abstract nature of conceptual system design, the **context** (or environment) of the **system-of-interest** should receive significant attention. By analyzing the context of the system-of-interest, the **stakeholders** can be identified. In turn, stakeholder **needs** can be identified. The stakeholder needs will give reasoning for both the **concerns** and the **problem** definition. These could have the form of initial requirements or functions to be fulfilled. As simulations are the main tool to give insight in dynamic behavior of the system, the problem should have some inherent **dynamicity** [Calvano and John, 2004]. Note that if the problem does not concern this dynamic complexity, simulations add little value.

System Model Definition

To establish an **architecture** it is key to reason from several **architecture viewpoints**. Relevant **architecture views** are discussed in a separate section. The views are described with **architecture models**. The various views together form the **architecture description**. Combining various views with a **mapping** leads to the definition of a **system model**. A single system model that fits the architecture description exactly is not required. By varying the mapping, many system models can be created. The process of generating these mappings is considered to be creative synthesis [Topper and Horner, 2013] or design space exploration. Because reasoning occurs at an abstract level, the design space exploration does not aim for an optimal solution, but rather aims to explore various ways to approach the problem.

Simulation Model Definition

In section 2.5 simulations were defined as the *"the implementation of a model in executable form or the execution of a model over time"* [National Research Council, 2002]. This means that a **simulation model** can be established by extending a system model with at least an **executable formalism** to the model, and/or a **time period**. Both approaches aim to quantify and qualify the **system behavior** and gain insight in its dynamicity. The formalism for a specific model is captured in a **model kind**. This model kind often concerns a specific **domain.**

Perform Experiments

The simulation model should already be created with the experiments in mind. So that when the simulation model has been created and validated, it can be used to perform these experiments. The experiments aim to uncover the **key system characteristics** that influence the system's behavior. For example a sensitivity analysis can be helpful in this regard.

Analyze Results

The key activity in this step is to contextualize, **visualize** and explain how the key system characteristics and the system behavior relates to the various system concerns that were identified earlier in the simulation study.

Validate and Communicate Results

Verification and **validation o**f results should happen throughout the simulation study [Sargent, 2013]. For example the system model should already be verified and validated after a first concept has been defined. This helps to ultimately represent the views of all stakeholders in the system model. Early validation also strengthens acceptance of the model and its outcomes in the later stages of the study [Law, 2014]. During the development of the system model, various views have been constructed using architecture models with their own formalisms. This means that each stakeholder, on its own, or as a **group** has access to those models of which the formalisms are understood by that group or individual [Lutters, 2001]. Ultimately, the goal is to give each stakeholder access to relevant information by offering stakeholders this particular information in **accessible views**. This leads to a system

architect being able to come to informed **design decisions** together with the stakeholders, as all have the required insight to reason about a particular decision. This means that all stakeholders have access to the **architecture rationale**, as they understand the various relations, or **correspondences**, that exist between system elements and how these relations impact the system's behavior. The correspondences are governed by **correspondence rules**.

4.3.3 Essential Views in Modeling and Simulation

In Chapter 3, the essential views in modeling and simulation during conceptual system design have been identified. These are the triad of a **functional view**, a **physical view** and a **quantification view** as recommended in [Bonnema, 2014] and the **operational view** that represents the system's behavior.

4.3.4 Reference Model

Many of the concepts that have been highlighted in the previous subsections are already represented in the [ISO/IEC/IEEE, 2011] conceptual model. Therefore, the choice was made to extend this conceptual model (see also Figure 2.2) with the concepts presented in this section. The resulting reference model can be seen in Figure 4.3.

Within the reference model, several groupings of concepts can be seen. At the top, more generic concepts are visible that deal with the overall development process. The left side focuses on the involved stakeholders, their goals and their access to information. The right side focuses on the views required to communicate simulations in conceptual system design and how they relate. Finally, the bottom side lists concepts that deal with the simulation itself. This reference model offers an overview of relevant concepts and can serve as a mental checklist both when communicating simulations and doing research in this area.

4.4 Conclusions

This chapter has considered the communication of models and simulations in conceptual system design in more detail. It can be concluded that currently little support is offered and an overview is lacking. To address the lack of generic approaches centralizing communication and conceptual system design, a simulation process was described. This simulation process emphasizes both the importance of communication and the distinction between problem and solution domain. Furthermore, four challenges were identified that help to understand the complications that arise when using simulations in conceptual system design. These challenges were linked to the described simulation process by providing a framework that details the process steps for each of these challenges. The chapter concluded with a reference model that gives insight in the relation between relevant concepts in communication of simulations and the existing architecture concepts of the conceptual architecture model in [ISO/IEC/IEEE, 2011].

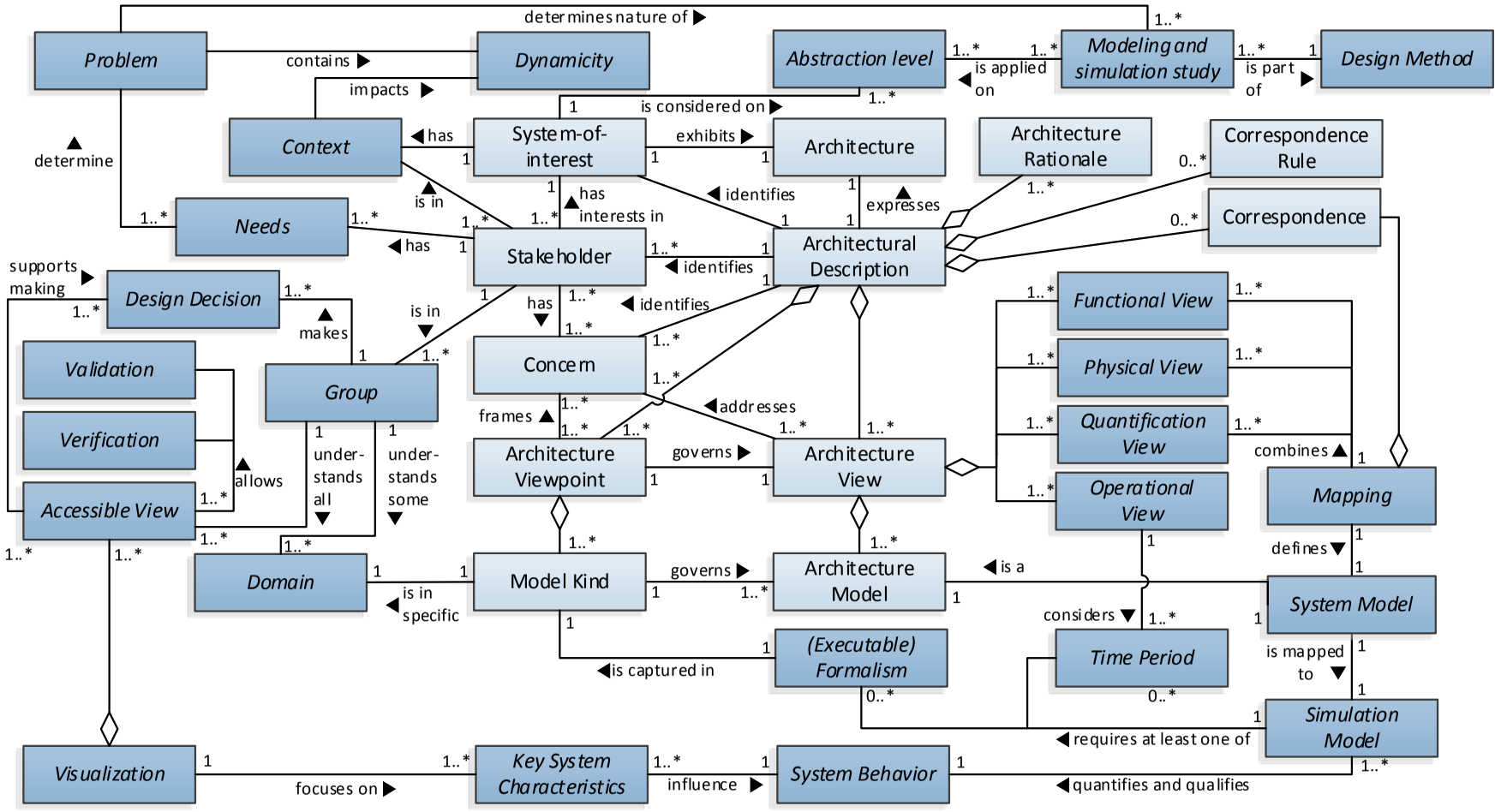
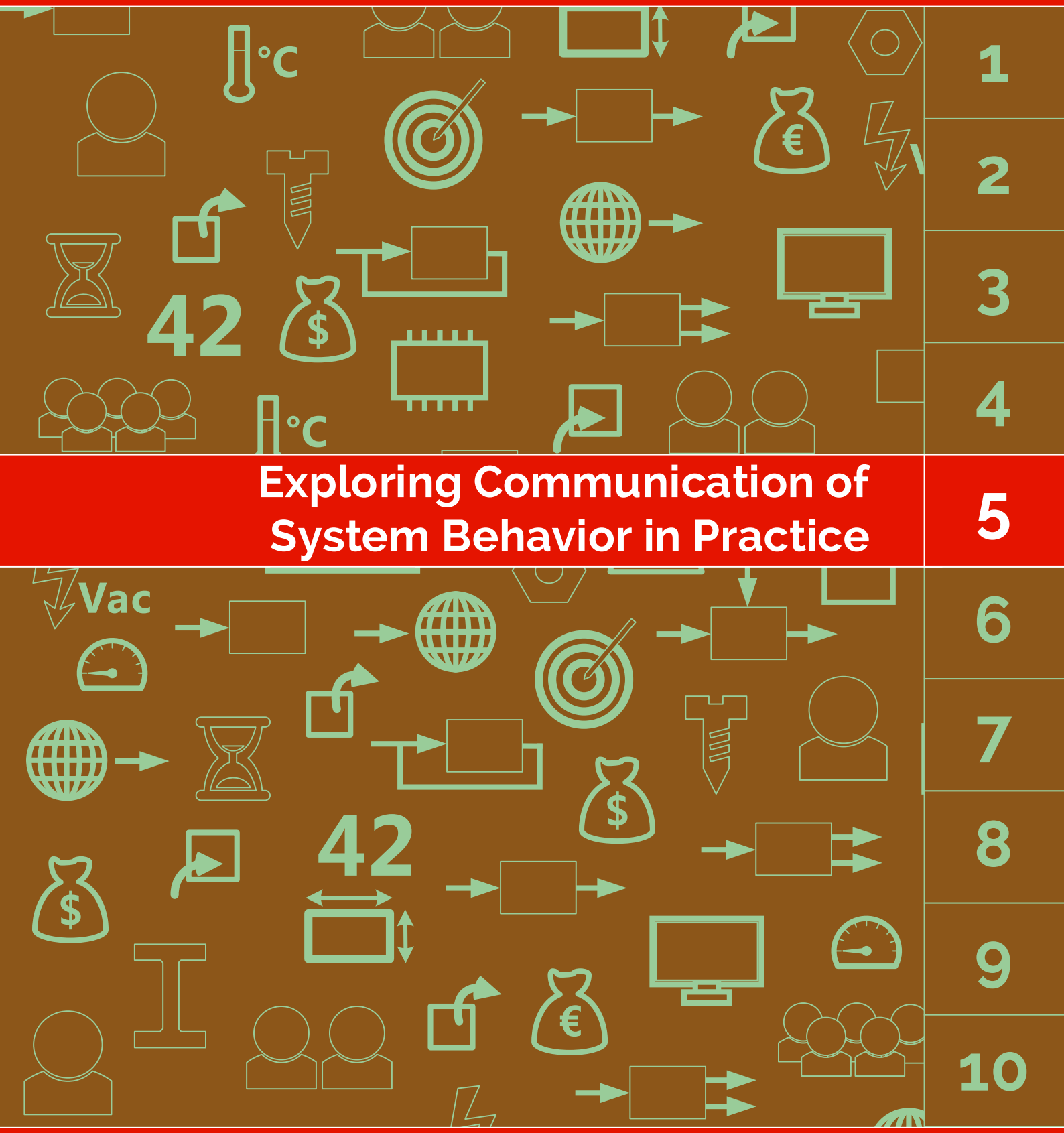


Figure 4.3 – Reference model describing key concepts in communication of models and simulations. The lighter boxes are the original concepts in [ISO/IEC/IEEE, 2011]



Exploring Communication of System Behavior in Practice

1
2
3
4
5
6
7
8
9
10

5

Exploring Communication of System Behavior in Practice

This chapter aims to answer the question of how communication of models and simulation in conceptual system design can be supported in practice. The chapter starts by describing requirements for the intended support in section 5.1. After that, this chapter examines the execution of two case studies aiming at identifying system behavior and subsequently communicating these behavioral analyses. The goal of these case studies is to identify parts of these processes that can be supported and to develop this support at the same time. The exploration in this chapter provides a basis for the method presented in Chapter 6.

The case studies presented in this chapter have been executed at the interventional X-ray department (iXR) of Philips Healthcare (see also section 1.3.1). An overview of the relevant concerns for an interventional X-ray system is shown in Figure 5.1, using the view presented in section 3.3.2. The case study discussed in section 5.2 focuses on the key driver hand-eye coordination, whereas the case study presented in section 5.3 focuses on a subsystem, the power distribution system. The case studies can be classified according to the types of design processes identified in section 3.1.2. The case study in section 5.2 is an initial study to a system-wide redesign, while the case study in section 5.3 is a subsystem redesign. The case study in section 5.2 preceded the case study in section 5.3. This second case study thus builds on the knowledge gained in the first case study.

5.1 Requirements for Intended Support

Before exploring how to support communication of models and simulation in conceptual system design, the requirements for the intended support [Blessing and Chakrabarti, 2009] have to be defined. In [Haveman and Bonnema, 2013], various requirements for high level models supporting design space explorations were discussed. Based on that discussion and

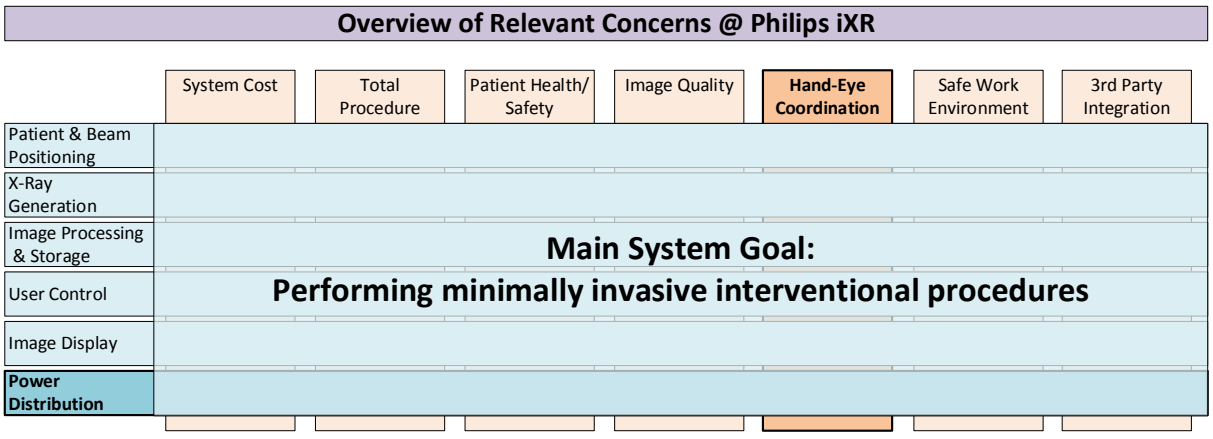


Figure 5.1 – Overview of relevant concerns for an interventional X-ray machine

Table 5.1 – Overview of main requirements for intended support in case study, adapted from [Haveman and Bonnema, 2013]

#	**Requirement for intended support and rationale for the requirement (in italic)**	**Related Simulation Process Step**
1	Support identification of relevant stakeholders, parameters and requirements; *needed to provide context for design problem*	Define Design Problem
2	Capture relevant parameters and requirements in an accessible overview; *necessary to define the problem clearly for all stakeholders, in order to create a shared view on the problem*	Define Design Problem
3	Enable definition of separate application and platform configurations and support mappings of applications on platforms; *the system model can be described by the views on application and platform and their respective mapping, so these elements should be supported [Basten et al., 2010]*	Define System Model
4	Support insight in the operational view or workflow; *this overview is important, especially regarding aspects such as safety and ease of use, as different use cases can be made explicit in this manner*	Define System Model
5	Support definition and adaptation of generic system behavior; *this supports uncovering the main behavioral characteristics and their influences on one another*	Construct and Test Simulation
6	Support an easy definition and adaptation of system models; *this allows exploration of various system models. [Morse et al., 2010]*	Perform Experiments
7	Support exploration and evaluation of system behavior; *this allows analysis of the simulation outcomes with respect to system behavior*	Extract and Analyze Results
8	Connect to captured requirements in definition steps; *solutions are evaluated based on their ability to meet the requirements. This means that the model should support a comparison of the two*	Extract and Analyze Results
9	Support communication of information across multiple disciplines; *design decisions are often taken by a group of stakeholders from multiple disciplines. It is critical that a model allows and supports communication of information across multiple disciplines [Do et al., 2012]*	Validate and Communicate Results
10	Support sense making of information by ensuring a high coupling of data; *to makes sense of the data presented, a high coupling between various data elements must be supported, as is discussed in [Heer and Agrawala, 2007]*	Validate and Communicate Results
11	Ensure a prominent place in communications for relevant key drivers and more detailed requirements; *giving these a prominent place will support engineers to reason about these aspects explicitly*	Validate and Communicate Results

the previous chapters, an overview of the main requirements for the intended support is given in Table 5.1. For several requirements, additional references are provided.

5.2 Behavioral Analysis of Hand-Eye Coordination

The overall research objective of this case study is to explore communication of models and simulations in conceptual system design. In order to do so, a design problem is required that is still in the conceptual system design stage and where applying a simulation analysis is useful. Within Philips iXR, a proposed redesign of the image processing chain offered this

possibility. In order to structure the execution of the case study, the overall objective is detailed in several concrete research goals.

First of all, the case study aims to complete a simulation analysis in the conceptual system design stage. This gives more insight in the identified challenges in Chapter 4 and allows for a review of whether these challenges can indeed be addressed. As an example, the approach is to use a highly abstracted system model and a generic simulation. Section 5.2.1 explores this research goal and describes how the simulation process is executed.

Second, the case study aims to deliver a concrete contribution for Philips iXR: to gain insight in the dynamic behavior of the image processing chain. The insight should be gained by both uncovering and qualifying existing dynamic relations in the system as well as through supporting exploration of various system models. The research goal is to investigate how to support this process. How this support is designed and implemented is described in section 5.2.2.

The third and final research goal of the case study is to investigate how to support the communication of the insight gained through the analysis with multiple stakeholders within the iXR department, such as system architects and engineers. Section 5.2.3 discusses this research goal by describing the communication process and support used during the case study.

While the process is described chronologically throughout this section, the actual process included numerous iterations and for example returns to the initial analysis phase. Furthermore, part of this case study has been published in [Haveman *et al.*, 2014].

5.2.1 Execution of the Simulation Study

To address the first research goal, a simulation study is conducted following the process described in Figure 4.1. This subsection describes how this process is executed. The last step of this process, validate and communicate the results, is treated separately in section 5.2.3. As the research goal is to review whether the identified challenges in section 4.2.1 can be addressed using the framework provided in Figure 4.2, each step will be reviewed based on this framework.

Design Problem

The first step is to define the design problem. In this case this concerns the fact that a physician relies on the images provided by a medical imaging system to coordinate his actions during interventional procedures. For example, a physician may need to move an inserted catheter through blood vessels of a patient towards the patient's heart. The physician does this by pushing and twisting the inserted catheter from the outside. If the time between a manipulation and the action shown on the display is too large, it is difficult to coordinate these actions. A top level function flow of this process is shown in Figure 5.2

The time it takes for an action to be shown on screen is termed the system latency. For many interventional procedures, a low latency (~150ms) is required to facilitate proper hand-eye coordination for a physician. When the case started, an excellent overview of system latency in the imaging chain was present due to the fact that knowledge on this subject was consolidated in an A3 Architecture Overview [Ouwerkerk and Canjels, 2011].

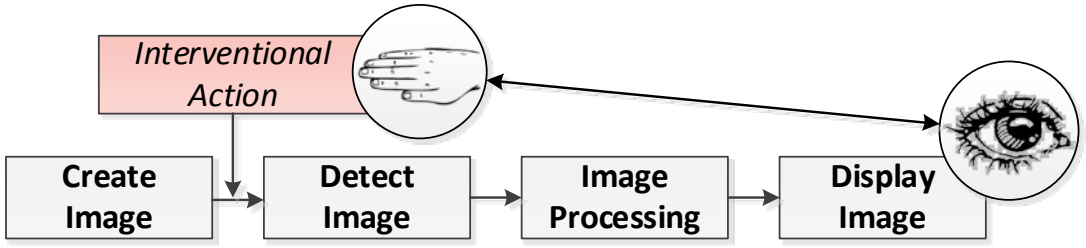


Figure 5.2 – Functional flow of the interventional imaging chain, emphasizing hand-eye coordination for a physician, adapted from [Ouwerkerk and Canjels, 2011]

Reasoning from this overview, a system architect showed that a redesign of the system could reduce the total latency and reduce complexity of the system. The estimation of the system architect was based on a summation of estimated latency times of components in the imaging chain.

However, the total time is not the only influence on hand-eye coordination, as the variation between subsequent image latencies influences the hand-eye coordination as well. This variation in system latency is termed jitter. Specific patterns in jitter cause it to be noticeable on screen; this visible jitter is termed a glitch. This is also visualized in the context diagram in Figure 5.3. Unfortunately, it is hard to gain insight in the cause of jitter as this cannot be analyzed with a simple summation. Furthermore, measurements on the real system are currently done with a polling rate of one second, which gives insight in average latency, but not in differences between subsequent images.

The process continues with an analysis of the definitions of system latency, jitter and

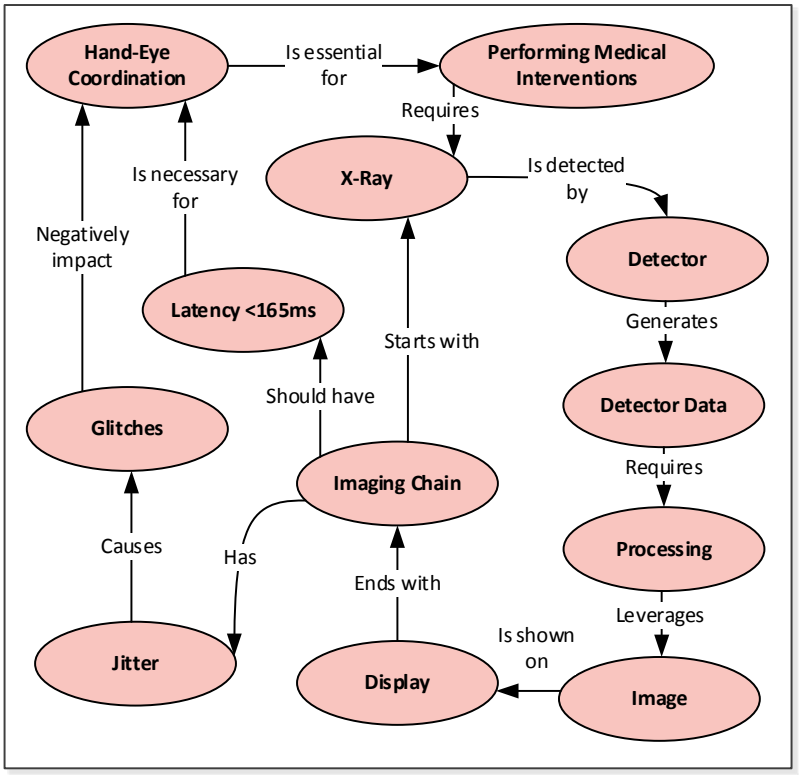


Figure 5.3 – Context Diagram of Hand-Eye Coordination in the iXR Imaging Chain

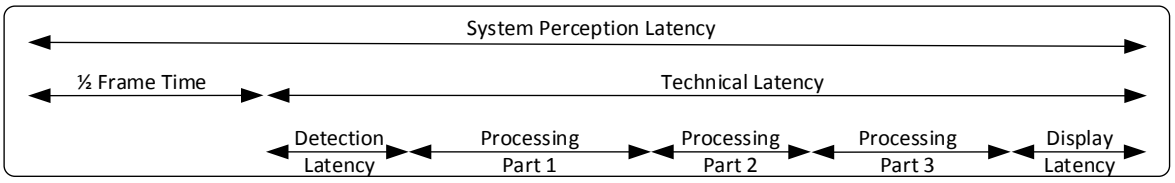


Figure 5.4 – Overview of latency components for iXR imaging chain

glitches. Figure 5.4 gives an overview of the decomposition of the latency throughout the system. An interventional action of a physician can start at any moment in between two successive x-ray pulses. Therefore, an average time of half of the frame time is assumed. The technical latency consists of various parts that correspond to the steps shown in Figure 5.2.

When a glitch occurs, the time between subsequently displayed images is fluctuating. Within iXR, a threshold value was determined through trials with physicians. This threshold value indicates the amount of variation that is still allowed. Furthermore, existing requirements for glitches were formulated by specifying maximum amounts of single and consecutive glitches within a defined number of images. The requirements are constructed in this way, because a single or a few glitches only impact user experience. Multiple glitches need to occur before the actual hand-eye coordination is affected, as is also indicated in Figure 5.3. Finally, while the exact definitions are confidential, it can be noted that especially the glitch definition was a central discussion topic throughout the study, as this definition determines the amount of glitches that are found during operation.

A review of this step based on the framework provided in Figure 4.2 shows that the focus has been on identifying stakeholders and key concerns, as well as questioning for example the glitch requirements to ensure an appropriate problem definition. Furthermore the focus has been on characterizing the context, which has been visualized in Figure 5.3.

System Model

The second step concerned the definition of the system model. In order to define a high level system model, the Y-chart modeling approach (see Figure 3.2) is used. The reason to aim for a high level model is to identify generic behavior that influences the jitter in the imaging chain in all configurations, not only in a very specific one. For example, when this model was discussed with engineers, imaging specialists and other persons within iXR, some engineers indicated that context switches in the memory of an image processing computer introduced some hold-ups. However, modeling this kind of behavior requires an extremely detailed model, which is especially focused on the architecture of the image processing computer in question and only identifies one specific cause of variation. In the conceptual system design stage, it is better to focus on a high level model that enables a full imaging chain analysis, coupling all functional steps into one model.

To create the Y-chart application view for the high level model, the functional flow in Figure 5.2 serves as a basis and is expanded to include various relevant functions, such as the transfer of data between functions. The functional steps in the sequential image processing chain can be seen in Figure 5.5 The same approach is used to model the platform view, which can also be seen in Figure 5.5. The platform view indicates the

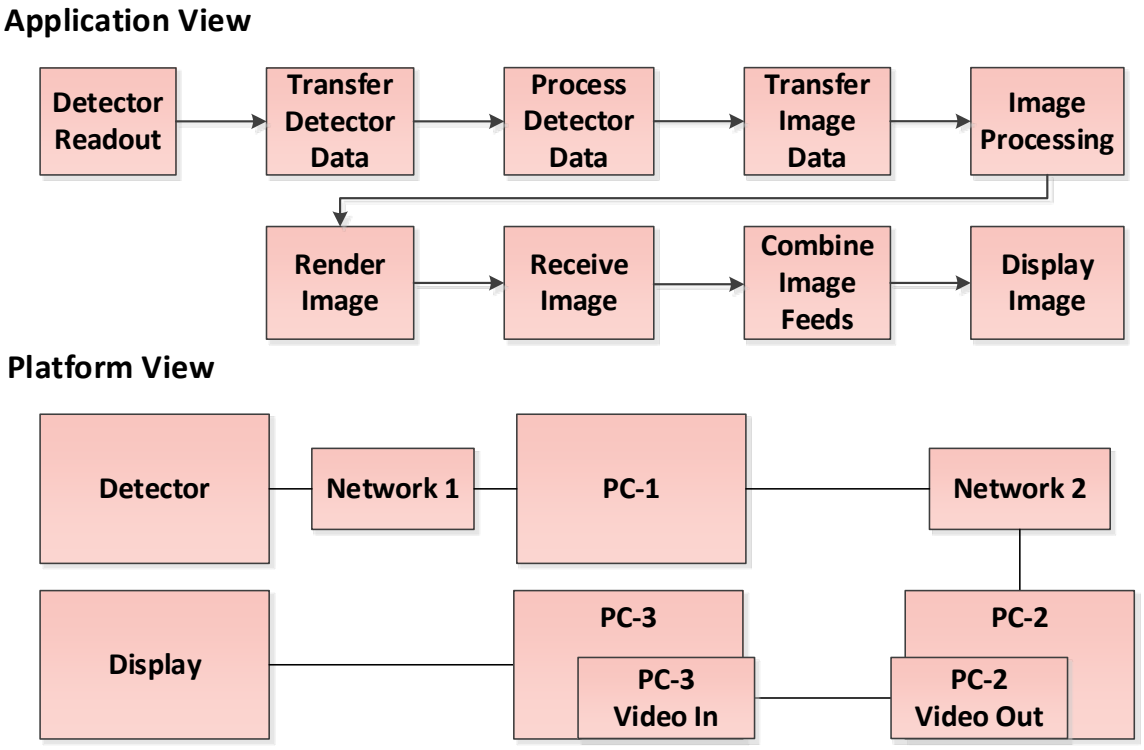


Figure 5.5 – Application view (top) and platform view (bottom) for the high level model. This is a non-confidential abstraction. The application view used in the case study contained 15 elements

components and the available connections between components. Some components are excluded from the model. For example, the connection between PC-3 and Display is a DVI connection. This connection was assessed to be insignificant at this abstraction level.

The quantification of the application view and platform view is done mainly by performing latency measurements on the existing system. These measurements consist of a execution time for a specific function on a specific platform. The execution times can be used to quantify the task load of the application component and the operation speed of the platform component. In a detailed model, it might be possible to exactly define the task load with a number of bytes and the operation speed as a number of bytes per second. A division of these two numbers results in the execution time. In this high level model, no additional information next to the measured system latencies was readily available. Therefore, both the load of an application element and the operation speed of the platform element are chosen in such a manner that division of the two resulted in a similar latency time as is measured.

The resulting system model including this quantification can be seen in Figure 5.6. In this figure, the numbers in the top right of each block indicate the relative operation speed of the platform element, where the numbers in the bottom right of the smaller blocks indicate the relative task load of the application component. Both are chosen in such a way that the resulting latency time is in milliseconds. For example, the duration of the task "process detector data" on "pc-1" is 2 milliseconds. The system model also includes the mapping of application elements to platform elements. In this model every platform element contains a single application element. The actual (confidential) model used in the

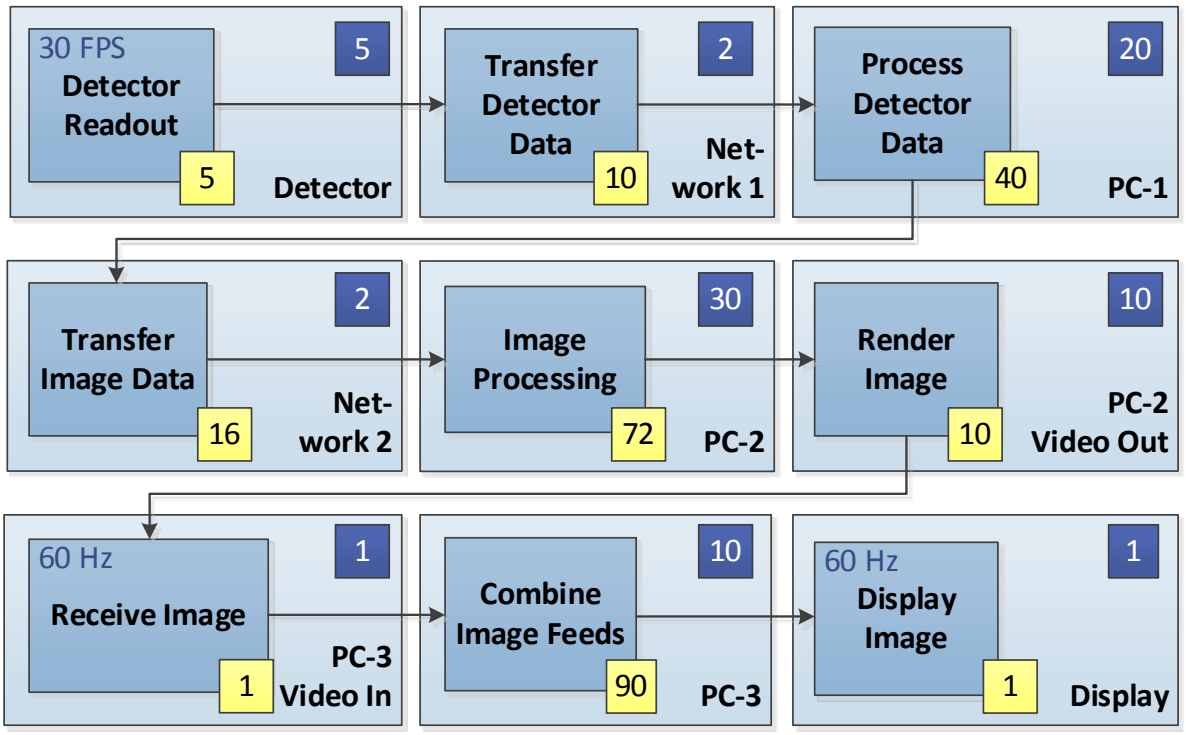


Figure 5.6 – System model with mapping of application functionality (darker boxes) to platform functionality (lighter boxes), including quantification of application and platform elements.

case contains fifteen application and nine platform elements. This confirms that the 1:1 mapping is optional, as multiple application elements were mapped to a single platform element. Finally, the system model also includes clock frequencies for some of the application elements. For example, the images enter the imaging chain with a frequency of 30 frames per second (FPS), while "display image" and "receive image" both operate at a frequency of 60 Hz.

During the development of the system model, various abstractions are made to realize the views required for the Y-chart modeling. Reflecting on this stage with the framework provided in Figure 4.2 shows that the approach to deal with uncertainty in this stage is to focus on the system as a black box and therefore ignore more detailed interactions such as context switching. Furthermore rudimentary latency data is used to quantify the system model.

Constructing and Testing the Simulation

After the definition of an initial system model, the third step is to construct and test the simulation. Therefore, the generic behavior so far is translated into a simulation model. In order to align to the existing validation procedures of the system that involve a test run of images on a prototype system, the model aims to simulate a given amount of subsequent images processed by the imaging chain. This number of images can be aligned to testing procedures, which measure system behavior for 2000 images in a single run, but can also be varied.

In order to attain a realistic simulation model, behaviors are analyzed for each of the application and platform components. Three types of application components are identified:

- *Regular*, this type of application component receives an image and immediately initiates its processing. After the processing has completed, the image is forwarded to the next application component. To characterize this process, only the load is designated.
- *Input*, the input application component generates images with a specified framerate. This process takes a certain time, which is also indicated with a load. When the processing is completed, the image is forwarded to the next application component.
- *Display*, the display application component checks for the arrival of a new image once per specified refresh cycle. A certain time is required to process this image, which is indicated with a load. The process either displays the newest available image, ending the imaging chain, or forwards the images to the next application component.

Additionally, two types of application components are identified, which are:

- *Processor (CPU)*, a processor from the CPU (central processing unit) type will process an image for a specific application element. The CPU processor has a certain operation speed, and will delay the image for a specified time indicated by the load of the specific application element divided by the operation speed of the CPU processor. While one image is processed, the CPU is unavailable for other images. Note that an abstraction was made from the use of multiple cores, which would allow some form of parallelism.
- *Processor (FPGA)*, a processor from the FPGA (field-programmable gate array) is a pipeline processor. This means that it operates in the same manner as the CPU processor. However, it accepts new images while the previous image is still being processed. It accepts these images every clock cycle of the FPGA.

In the simulation model, the included parameters can be considered in different ways. For example, a parameter can always have a given value and cannot be changed, or the possibility can be given to vary the parameter. This variation can be done between simulation runs or within a single simulation run. If the variation exists within a single simulation run, this can be approached in different ways [Law, 2014]. For example, a maximum and minimum value can be defined and a random number between those values is selected each time. It is also possible to choose distributions other than this uniform distribution, for example a normal distribution.

Latency measurements on the existing system show that there is variation in the execution times of the imaging system. A uniform distribution is chosen to represent this variation. Furthermore, as is mentioned earlier, engineers mentioned that context switches in the memory of certain PCs sometimes caused delays. These delays are represented with a given chance of occurring and a given delay length for a platform element.

Reflecting on this step, it can be seen that the simulation model was developed to align to existing validation procedures and models generic behavior as advocated in the framework in Figure 4.2. Furthermore, the way in which parameters were defined and can be adjusted was discussed.

Experimentation

Once an initial implementation (see also section 5.2.2) of the system model and simulation model was established, experimentation with the system model could be started. During

this phase, both the system model and simulation model were updated a number of times to accommodate for changes in the system model and simulation model behavior.

The starting system models that are simulated are based on the model shown in Figure 5.6. Detailed results of this stage can be found in appendix B.2.1. Simulating the model from Figure 5.6 resulted in a constant latency and no jitter, which is to be expected, since no variation was introduced (see also Figure B.6). However, when a variation of 10% in PC-2 and PC-3 was introduced, the resulting latency was still constant and jitter was still absent. This behavior was not expected by domain specialists as glitches already occur with small variations, but the model did not reflect that. The engineers also reported seeing a saw-tooth like profile in latency measurements of the actual system. After a further analysis it was found out that the obtained value of 60Hz for the display refresh rates was an abstraction made by an engineer, as the actual refresh rate was actually 59,9Hz. Simulations where both the clock was adjusted to 59,9Hz and variation was introduced indeed showed the behavior that was expected as well as the influence of jitter, resulting in glitches. A detailed analysis of the cause of these glitches can also be found in appendix B.2.1.

The experimentation continued with an analysis of the influence of the size of the variation in PC-2 and PC-3, given that the clocks of the display components were running on 59,9Hz as well. The detailed results of this stage can be found in Appendix B.2.2. An interesting observation is that sometimes, a simulation run produces an abnormal amount of glitches. The graphs of a regular simulation run and a run with this abnormal amount of glitches are shown in Figure 5.7. These graphs show the overall latency behavior. Glitches

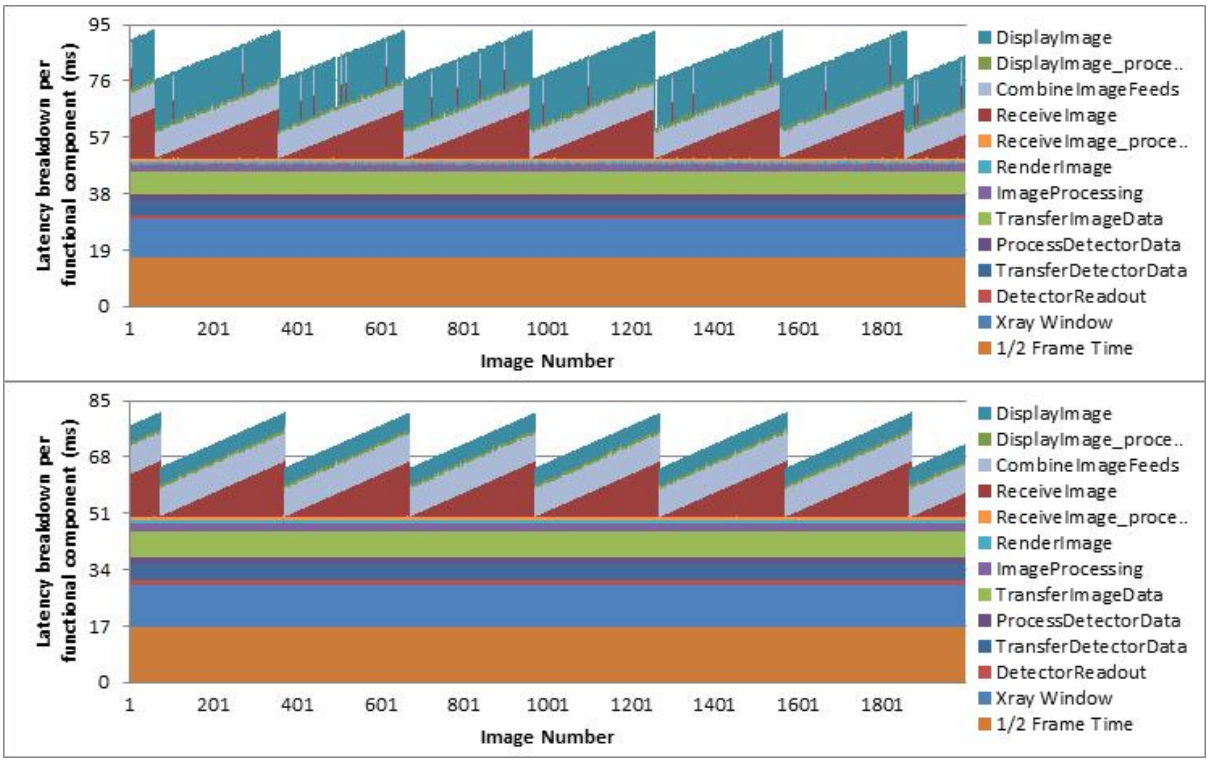


Figure 5.7 – (Top) Run with 10% variance, resulting in 2 glitches, (Bottom) Run with 10% variance, resulting in 176 glitches, the "outlier" case

Figure 5.8 – Clocks present in the system model

cannot be inferred directly from this graph, as they have to be calculated according to the definition. However, jitter, as well as the saw-tooth behavior can be observed. The tool described in section 5.2.2 and appendix B.1 allows a user to zoom in on these graphs.

Reflecting on this stage, it can be seen that the framework in Figure 4.2 states *"sensitivity analysis is key"*. It clearly showed that varying parameters, even those that were expected to be constant, influenced the system behavior. The experimentation further focused on determining the impact of varying the different parameters on the behavior of the imaging chain. Finally, the experimentation continued until the simulation model resulted in behavior that matched the expectations of stakeholders. At this point, the model was considered to be sufficiently reliable.

Extraction and Analysis of Results

As extending the sensitivity analysis did not result in any new findings, the experimentation phase was ended and the extraction and analysis of results was started. A major focus of this phase is to determine the cause of the occasionally large amount of glitches (which have been dubbed 'outlier' cases). Initially, no clear explanation for these cases could be found. Also, the fact that this behavior does not occur in the machines operating in hospitals reduced the models validity. Various model iterations, adaptations and additional timing analyses have been conducted to find the cause of this behavior. Eventually, the cause for the behavior was identified in the fact that the last two display components operated with the same refresh rate, as is also illustrated in Figure 5.8.

If two subsequent clocks have aligned periods, every image processed by the system arrives around the same time at the last of those two clocks. If this arrival moment is close to the time the display component refreshes, many glitches can occur. This is also shown in Figure 5.9. It can be seen here, that if there is variation in the arrival time of an image, it is possible for an image to either arrive from "receive image" just before or just after the start of the "display image" period. This results in two images arriving in the same period, leading

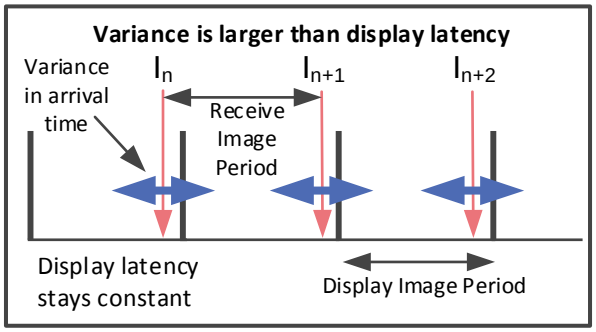


Figure 5.9 – An image (I) always arrives at the same time near a display refresh moment

to a non-display of one of these two images, as only one image can be displayed.

The reason that this does not occur in every run is based on the random start times of both display components. The behavior only occurs if the start time of the "receive image" clock plus the time it takes for an image to get to the "display image" component is similar to the start time of the "display image" clock plus an integer number of periods.

The modelled outlier behavior does not occur in the field, because once again it was determined that an approximation was used to model the clock periods. In reality, the clocks do not exactly have 59,9Hz periods, but vary depending on for example variance in the manufacturing process and operating temperature. However, the modelled behavior could theoretically occur, because there is a possibility that two display components with similar periods end up in the same system.

Furthermore, within Philips iXR, aligning all system clocks by physically linking them to get rid of the saw tooth pattern in the latency profile has been discussed. This behavioral analysis shows that in that case, there is an even larger risk of these outlier cases, if start times of display components are not chosen in a smart way. Another finding is that the glitch definition could actually be changed as the perceived situation for which the requirement had been defined was not reflective of the actual situation. The actual situation allows a larger amount of variance and the glitch requirement is thus too tight.

A review of this phase in the simulation process shows that the analysis focuses on explaining and visualizing the root cause of the behavioral characteristics. This is done using visual aids and by referring back to the basic concepts that make up the design, instead of focusing on a single system model. This aligns with what is advocated with respect to this phase in the framework presented in Figure 4.2.

5.2.2 Supporting the Simulation Process

The second research goal for this case study is to research and develop a support to execute the simulation process. One of the most important requirements is to allow the envisioned users, in this case mainly system architects, to easily generate variations of the defined system model, as well as evaluate the results of these models (requirement 6 and 7 in Table 5.1). This section discusses in more detail how these requirements were addressed in this particular case study. Additional information regarding the developed support can be found in appendix B.1.

The solution to these requirements is driven by the fact that system architects should be comfortable using the support and the outcomes. A logical choice is to use a support tool that system architects are familiar with. Therefore, the system model definition is embedded in Excel [Microsoft, 2010a], which is the tool that is most often used during these kinds of analyses. Using a custom software tool is possible as well, but it was deemed that this could raise the threshold to use the support tool. The implementation in Excel makes use of VBA (Visual Basic for Applications).

Excel can be used for the simulations as well, as it supports randomization required to do for example Monte Carlo simulations. However, it is infeasible to use Excel to model and execute the behavior described in the previous section for the application and platform components. The behavior described requires tooling that supports discrete event

simulation [Robinson, 2008b]. Examples of discrete event approaches are Analytical Software Design (ASD) [Broadfoot, 2005], Colored Petri-Nets (CPN) [Wang and Dagli, 2011] or POOSL [Putten and Voeten, 1997]. However, ASD does not support simulations [Hooman *et al.*, 2012] and is used more for formal verification. [Hooman *et al.*, 2012] also note that POOSL is more suited to gain initial insight in the structure and interaction between components. [Wang and Dagli, 2011] show an application to simulate system models defined in SysML with CPN. POOSL was chosen to implement the simulation model. Possibly using CPN's or similar methods would have allowed similar results. However, expertise and example models were readily available for POOSL, as well as example applications within Philips iXR.

An overview of the resulting implementation can be seen in Figure 5.10. The high level system model allows a user to configure various versions of the system model using a system configurator. The user can edit parameters, add or remove application and platform components and change the mapping. The user can select one or several of these configurations, indicate the number of images for each run and start a batch of simulation runs. A parser was written to convert the system configuration defined in Excel VBA into POOSL compatible files. Using a batch file script, these POOSL files are run in the dedicated high speed POOSL simulation engine, Rotalumis. The output of the simulation runs are text files containing an image number and time stamps of the image entering and exiting all application components. Excel imports and parses these text files and calculates key parameters, such as the average latency, the maximum latency and the number of glitches that occurred in the simulation run. Within Excel, a simulation batch can be loaded and the results for different runs within the batch can be compared. Detailed results for

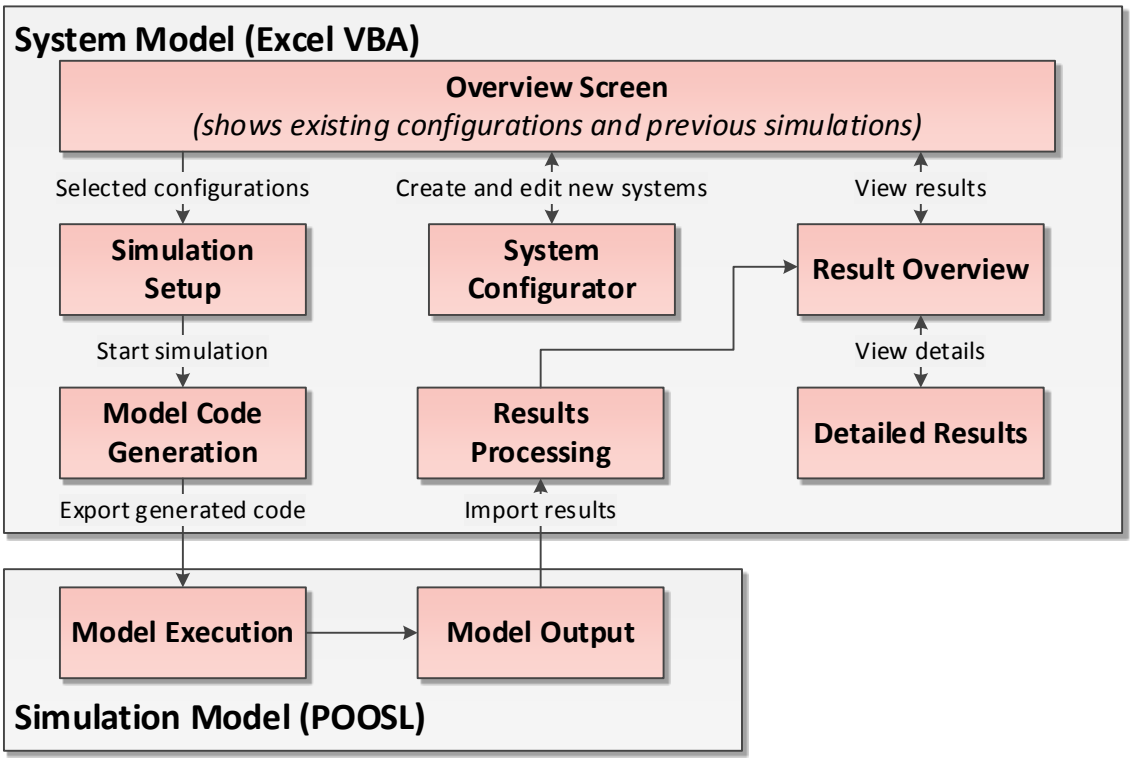


Figure 5.10 – Overview of implementation of high level system model and simulation model

each simulation run are available as well. These show graphs detailing the latency per image and give a latency breakdown per application component, as well as detailed metrics. Appendix B.1 gives a detailed description of the user interface of the system model.

This subsection described the implementation of a support, mainly aimed at requirement 6 and 7 in Table 5.1. The resulting support allows an easy definition of iXR imaging chain system models, as well as direct simulation of these models. The results can be analyzed through the support as well. Therefore, these requirements were met in the developed support.

5.2.3 Supporting Communication

This subsection focuses on the third research goal of this case study, which is the communication of insight gained through the analysis with multiple stakeholders within the iXR department, such as system architects and engineers.

As already mentioned, various iterations were used throughout the analysis. In each of these iterations, the results or inner workings of the model were explained and discussed with domain specialists. Initially, a standard report style and presentations were used to document and communicate the analysis. However, discussions relied on linking certain pieces of information that were spread throughout the report and presentations. Especially as these grew in size, meetings became a "scrolling exercise" when using the digital versions, or a large pile of documents when using printed versions.

The author of this thesis was already familiar with A3 Architecture Overviews [Borches, 2010], which are an efficient communication tool and provide a good system overview. However, only the model side is utilized as this side is used mostly for communication. The template for the model side of an A3 Architecture Overview can be seen in Figure 5.11.

The research focused on whether an A3 Architecture Overview could fulfil the communication role required. This means supporting communication without losing overview

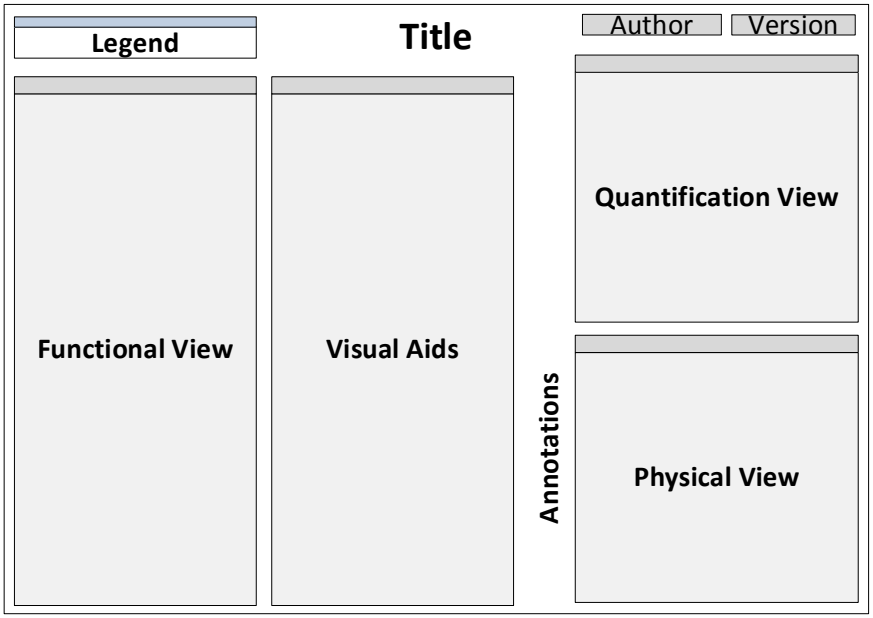


Figure 5.11 – A3 Architecture Overview - Model side template [Borches, 2010]

because of digital overviews with many pages or slides, or having an information overload because of a large amount of printed media. However, to be able to use an A3AO, this template had to be adapted. Several important concepts (see Figure 4.3) are not represented in a classic A3AO. These are for example an explicit mention of the goals of the modeling activity and the validation of the model. Furthermore, in this analysis the system was represented with a Y-chart model, which is a mapping of the functional view to the physical view, with additional quantification.

The A3AO forces a creator to select only the most relevant information for the topic at hand due to the constraints in size and readability. In this case, this was quite similar and the communication was centered on the goals of the modeling activity. The main goal of the case study is to give insight in the main issues affecting jitter in the imaging chain. Therefore, these issues are introduced and explained explicitly. Furthermore, the presentation of simulation results focuses on these issues. The structure (or template) of the resulting A3AO can be seen in Figure 5.12. Here it can be seen that blocks are designated for the goals of the simulation analysis, as well as ample space for the validation of the model. Visual aids are used throughout the domain introduction, and in other blocks when required to explain certain concepts. In a normal A3AO, design concerns or decisions are listed and labelled with colored stars. These stars are placed at locations in other views that link to the same key concern. In this case, the conclusions and recommendations are labelled with stars, to provide a link to the argumentation for a certain conclusion or recommendation. The final resulting A3AO can be viewed in Appendix H.1.

This A3 Architecture Overview was used during design discussions in the later stages of the modeling and simulation analysis. Users liked the visual aids explaining various concepts throughout the A3AO, but indicated that the contents were still complex and the overview felt too full. Especially when looking at the simulation results block (see Figure 5.13), it

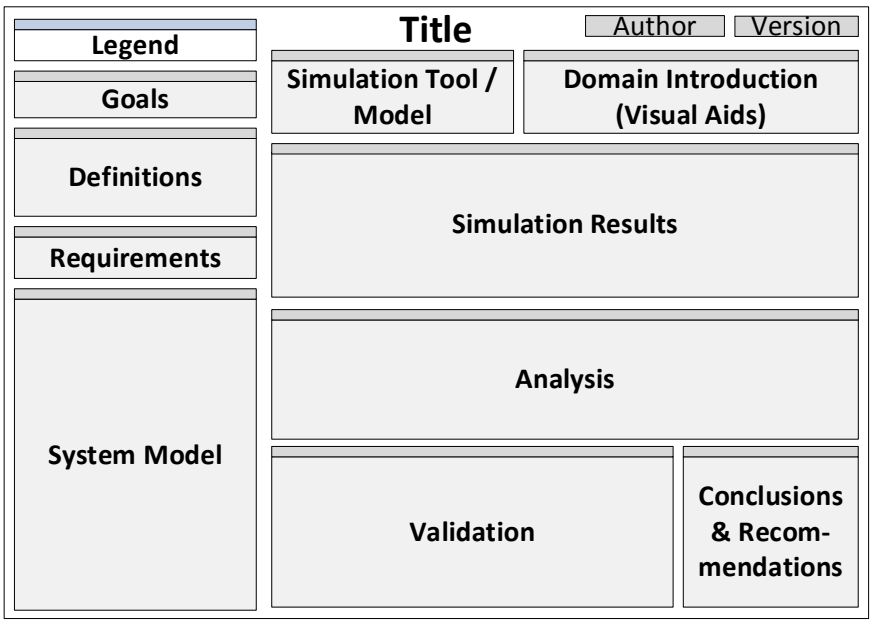


Figure 5.12 – A3AO template used to communicate latency & jitter analysis, a full size version with content is available in Appendix H.1

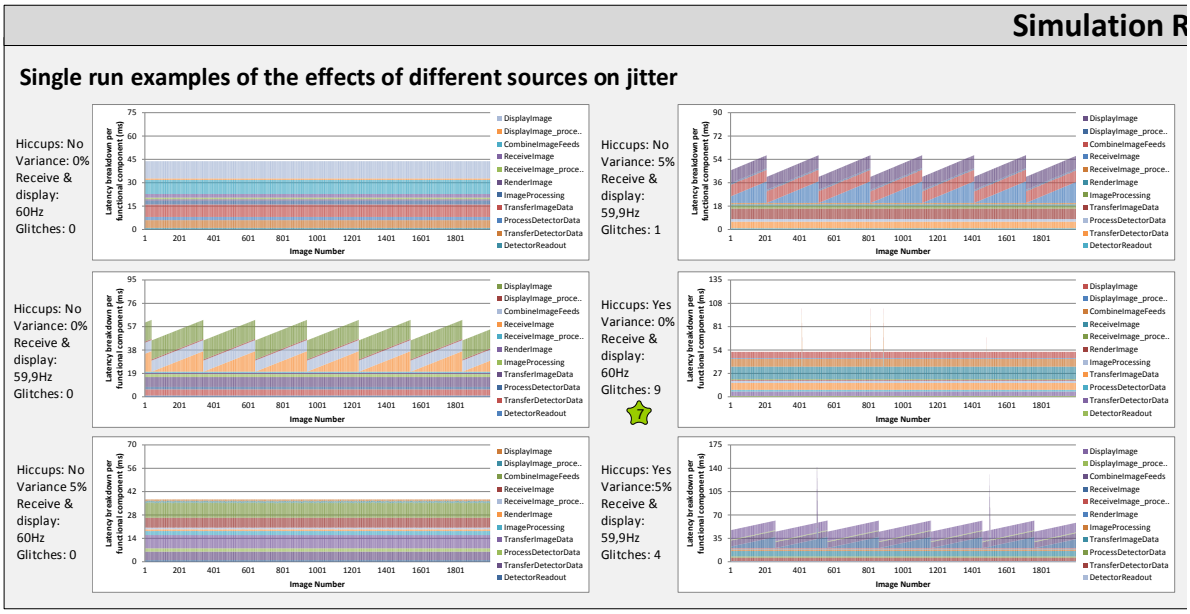


Figure 5.13 – Left side of simulation results block of A3AO used to communicate the modeling and simulation activity. It can also be found in the full-size A3AO in Appendix H.1.

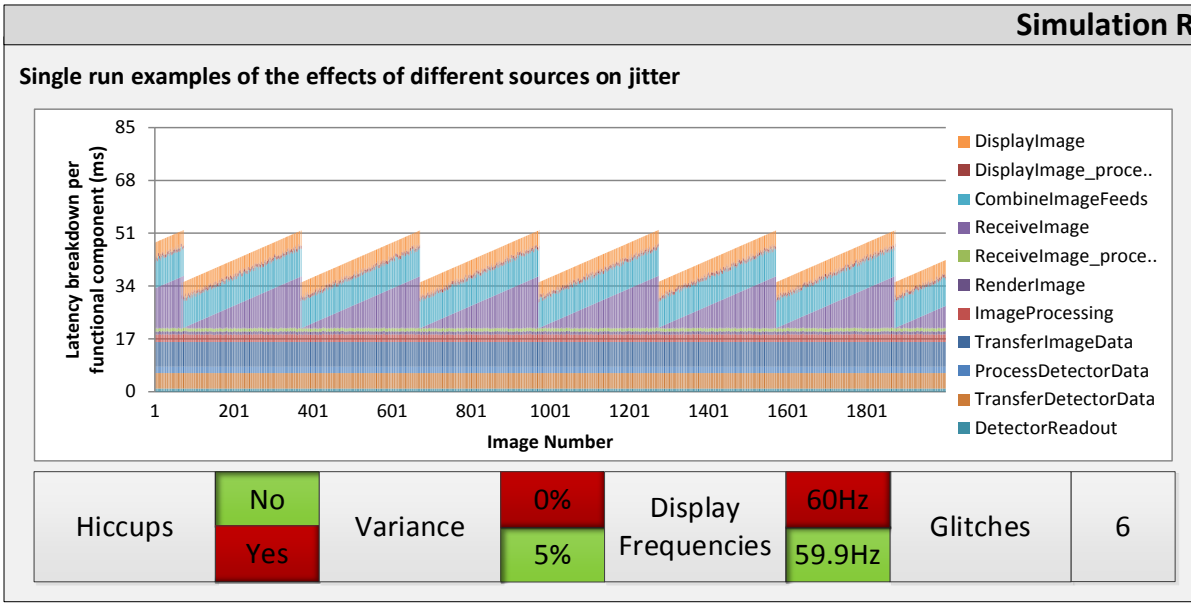


Figure 5.14 – Left side of simulation results block with user interface

can easily be deduced why users gave this feedback. Of course, Figure 5.13 is hardly readable resized to the paper format of this thesis, but even when printed on A3 paper size, readability only improves a little and meaningful information is hard to distinguish.

At the end of the study, the user feedback regarding the fullness of the A3AO prompted research to encapsulate some of these visualizations better. In the end, the six graphs presented in the left side of the simulation results block in Figure 5.13 were encapsulated in a small user interface inside the A3AO. This interface allows a user to browse through various simulation results. The user interface can be seen in Figure 5.14 and allows a user to both turn the identified sources of jitter on or off and explore the effects on the

Figure 5.15 – A3AO with user interface embedded in the simulation block being used on a tablet

simulation results. Figure 5.14 shows the user interface with results for a simulation run that has set hiccups to disabled, variance to 5% and display frequencies to 59,9Hz, resulting in 6 glitches.

The drawback of creating such a user interface is of course that the A3AO needs to be used digitally. In order to enable this, an A3-sized tablet was acquired. This tablet, a Dell XPS-18 [Dell Corporation, 2015], is actually a fairly light-weight and touchscreen enabled all-in-one pc running Windows 7. An example of its usage can be seen in Figure 5.15.

Within this simulation study, the user interface and tablet were not used as a resource and served solely as a demonstration tool for following cases. Initial reactions based on these demonstrations were positive, as potential users could see benefits of using the A3AO this way, with as main benefit the ability to explore more situations than a paper-based A3AO allows.

5.2.4 Discussion

The previous subsections have discussed the case study from the perspective of the three research goals. This subsection reflects on those goals.

The first research goal was to execute a simulation study in the conceptual system design stage. The description of the simulation process is similar to existing descriptions of simulation studies. However, the fact that the full imaging chain was simulated at an abstract level was new in the context of Philips iXR. A similar, recent approach is presented by [Hendriks *et al.*, 2014]. The quantification of the model was mainly based on measurements. However a lot of architectural knowledge can be inferred using only these measurements as was shown in [Berg *et al.*, 2015, Haveman *et al.*, 2014].

The second research goal was to support an easy high level system definition and subsequent exploration of system behavior. To this end, Excel was used as a front-end and the user interface in Excel VBA described in appendix B.1 was developed. While the developed support succeeded in allowing users to quickly define system models, it is not significantly different from existing simulation supports.

The third research goal, supporting the communication, proved to the most complex part of performing the simulation study. In this case study, this issue was addressed by developing an A3 Architecture Overview. After its creation, the A3AO served as a discussion tool during the last stages of the simulation study. Users perceived this A3AO to be a useful tool, but did comment on the fact that the A3AO was overcrowded. While it is normally already a challenge to avoid too much information on an A3AO when communicating mainly about static system properties, it was experienced in this case study that the different outcomes and behaviors of the system in a modeling and simulation study provided an even greater challenge. The creation of an interactive, digital version of the A3 Architecture Overview already showed some usefulness in this regard, but additional research is necessary to explore this issue further.

5.3 Communicating the Behavior of a Power Distribution Subsystem

The second case study concerns the redesign of the power distribution subsystem of the Philips iXR Allura system. This power distribution subsystem does not only provide power to the system, but also manages start-up and shutdown scenarios. Within this redesign project, a simulation model had been developed by a software engineer to formalize and test the system's behavior [Schuts and Hooman, 2015b]. However, the model was only used for this analysis and not for communication with other stakeholders in the project.

This led to the definition of the main objective of this case study, which is to leverage the knowledge contained in the simulation model to other stakeholders and to be able to have worthwhile design discussions based on the insight gained. The associated research goals are to (i) identify the relevant knowledge that needed to be communicated and (ii) support the communication of this knowledge.

First, the topic of this case study is further introduced in section 5.3.1. The first research goal is addressed in section 5.3.2. The second research goal is addressed in both section 5.3.3 and section 5.3.4, which respectively address the implementation and usage of the communication support. The section concludes with a discussion in section 5.3.5.

5.3.1 Introduction

The redesign of the power distribution subsystem is driven by a possibility to reduce complexity and costs of this subsystem, as well as to improve interactions for various stakeholders such as the service department.

The project was already running for a few months when this case study was started. At the start of the case study, requirements documents and design documents had already been created, as per the normal way of working within Philips iXR. These specifications were used as resource for presentations at suppliers. Furthermore, various drawings and schematics were also created and used within these presentations. Finally, a model was created [Schuts and Hooman, 2015b] to convert legacy configuration files to a domain specific language (DSL) based specification. This DSL specification allows the generation of a POOSL model. This POOSL model is used as an analysis model that simulates the complete start-up and shutdown behavior of the subsystem as a state machine. The user interface of this simulation can be seen in Figure 5.16. The left side shows the overall

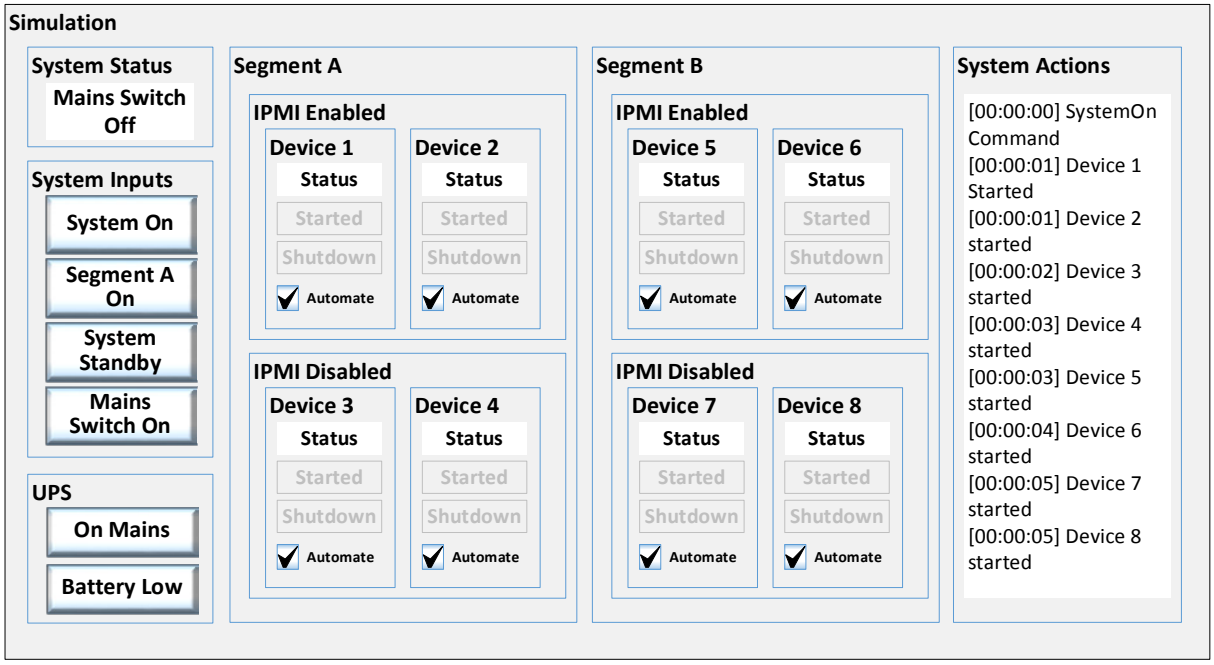


Figure 5.16 – User interface used to control simulation [Schuts and Hooman, 2015b]

system state and consists of buttons that control the system input. The middle part represents system devices, which show their status and can be controlled automatically or manually. The right side shows internal system messages.

This simulation based analysis proved to be useful for the creator, who is also the software engineer responsible for the design of the behavior of the power management subsystem. However, while the information contained in the simulation is valuable, a way to convey this information to other stakeholders is lacking.

5.3.2 Identifying Knowledge to be Communicated

The case study is started by surveying the available knowledge and documentation. This showed that it was unclear which knowledge was crucial in communication about the behavior of the subsystem. This is why one of the research goals is to identify the relevant knowledge that needed to be communicated.

To identify the relevant knowledge, the initial step is to create a system overview in the form of an A3 Architecture Overview. Selecting the information for this A3AO is done with the underlying goal of communication of the system behavior in mind. The structure of the resulting A3AO can be seen in Figure 5.17, whereas the A3AO itself can be found in appendix H.2. The structure of the A3AO shown in Figure 5.17 follows the guidelines of a regular A3AO. The only adaptations are the inclusion of a context view visualized as a systemigram and the reservation of a blank space for use cases. The context view is added to be able to structure important concepts in the subsystem and to show the connection to broader system concepts. A blank space for use cases is added, because this space is reserved for controlling the simulation. Through this simulation, various use cases can be executed and the operational view can be explored.

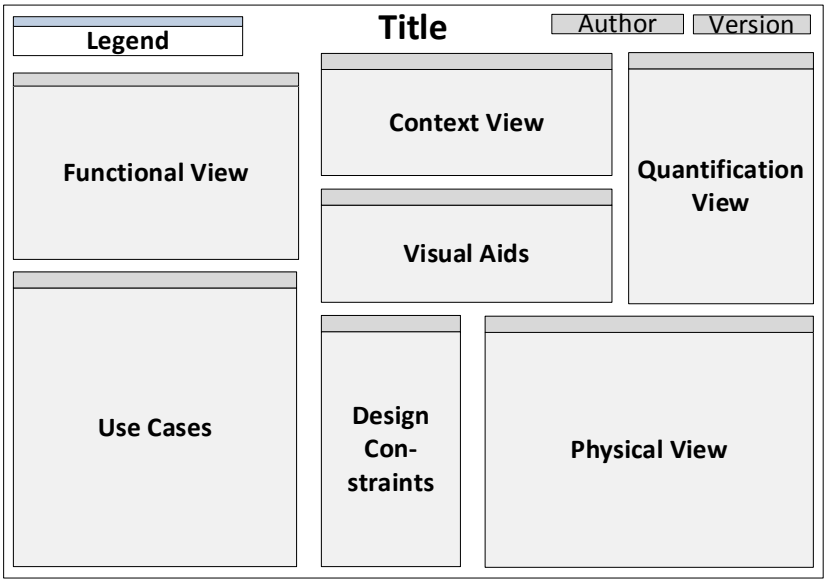


Figure 5.17 – Structure of A3 Architecture Overview used to create system overview of the power management subsystem, the full size version with content can be found in appendix H.2

To be able to explore this operational view, various choices could be made as the operational view can be considered from various perspectives. The user interface shown in Figure 5.16 represents the software developers' perspective. It shows the operation of the power management subsystem in terms of device statuses and signals being transferred between devices. However, to be able to communicate a system to stakeholders from multiple disciplines, other views are required as well. As [Bonnema, 2014] argues, a system can be represented well with a functional, physical and quantification view. The assumption is that the same holds for communication of the system's behavior. Therefore, it was decided to show the behavior of a system across these views, as is also argued by [Kruchten, 1995].

To identify the relevant knowledge to be communicated regarding the system behavior, the behavior is characterized by identifying relevant system states. The identified states are listed in Figure 5.18. In the actual case study, 24 states were identified, but for confidentiality reasons, this has been abstracted to 12 states in this thesis. Six main states are identified. However, several of these states are further detailed because power in those

1	Mains Switch Off	
2	Starting System	On Mains
3	Starting System	On Battery
4	System On	On Mains
5	System On	On Battery
6	System Standby	On Mains
7	System Standby	On Battery
8	Segment A On	On Mains
9	Segment A On	On Battery
10	Shutting Down	On Mains
11	Shutting Down	On Battery
12	Shutting Down	Battery Low

Figure 5.18 – List of states of the power management subsystem

states can be provided via either the mains of the hospital or via a UPS battery. If the battery level becomes too low, the system will automatically shut down. Furthermore, as also could be deduced from the user interface presented in Figure 5.16, there is the possibility to only power Segment A of the system, next to powering the complete system. In addition to this list of states, a state transition diagram was also created to ensure correct reasoning on the available states.

Even though the states and state transitions were already embedded in the POOSL simulation, they were not made explicit. By listing the states, it is easier to characterize the system's behavior as the system overview can be detailed for each state.

The next step is to consider each of the views in the A3 Architecture Overview in Figure 5.17 (or appendix H.2) for each of the states. Detailing these states is done by considering the system overview for each state and identifying which aspects of the system overview should be adapted to represent that system state. The relevant views to detail for each state are the functional, context and physical view. For example, the quantification view listing the key parameters is constructed in such a way that it only concerns generic parameters that do not change with the system state. Different quantification views can be used as well, that could list other parameters that do change with the system state. However, while creating the A3 Architecture Overview it was determined that the parameters that are currently listed are most relevant and there was no foreseen added value in showing behavior in this view.

A full overview of all views detailed for each system state can be found in Appendix C. An example of detailing the physical view for various system states is given in Figure 5.19 and Figure 5.20. The physical view shows which components are powered in which state and how the power flows through the system. Furthermore, the LEDs in the user interface module (UIM) represent the state of the system as well. The location view shows which

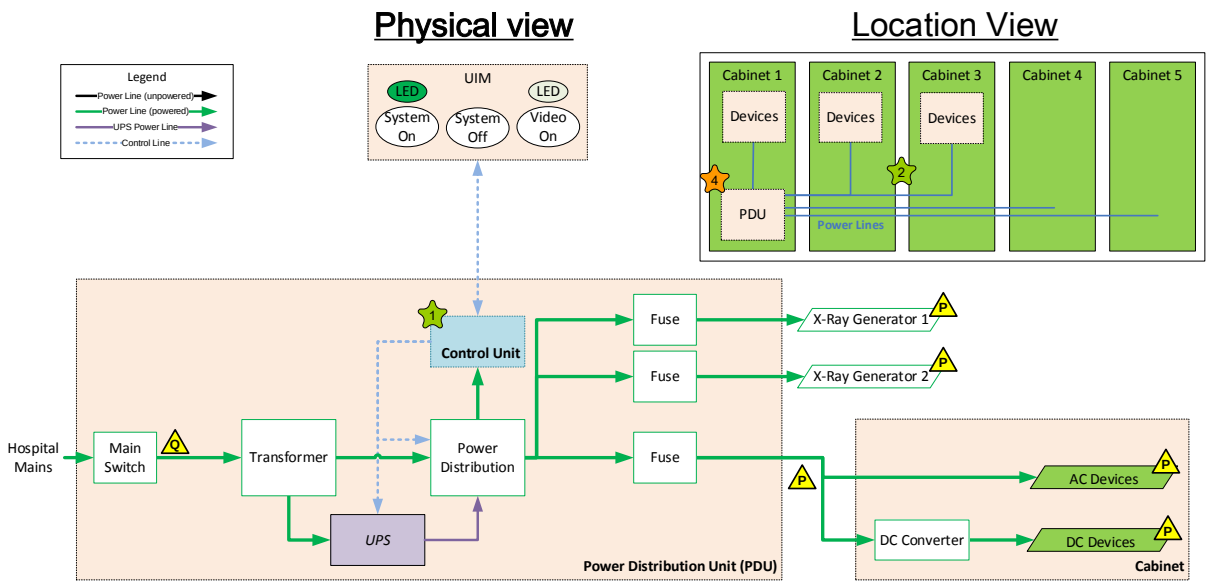


Figure 5.19 – Physical view for the three following states: "Starting System On Mains", "System On On Mains" and "Shutting Down On Mains"

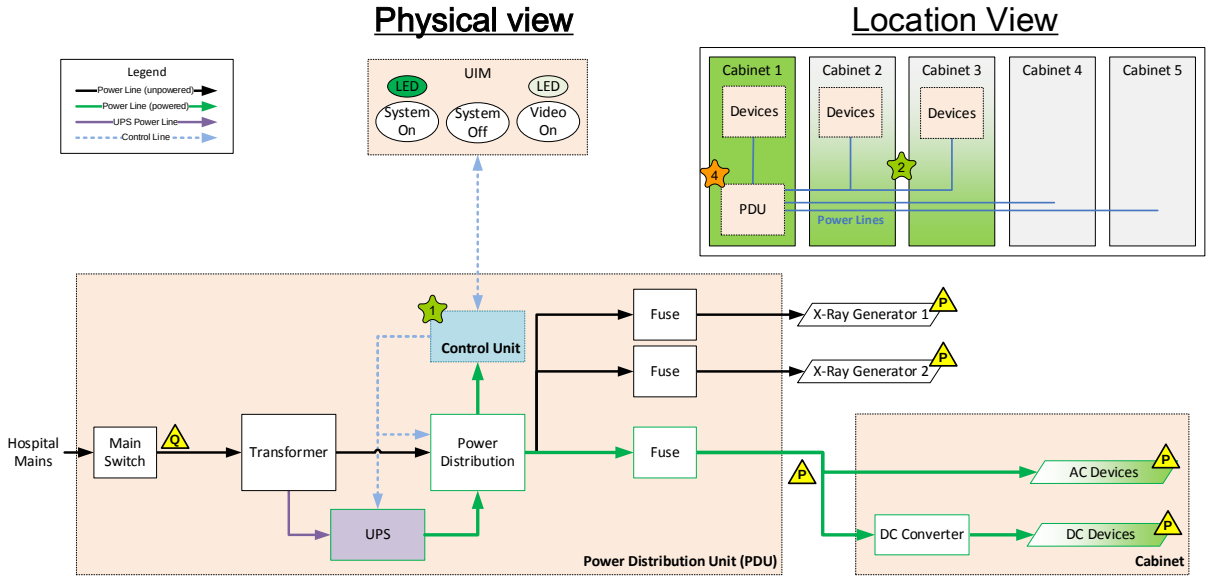


Figure 5.20 – Physical view for the four following states: "Starting System On Battery", "System On On Battery", "Shutting Down On Battery" and "Shutting Down Battery Low"

cabinets are powered. A full green box means that the cabinet is fully powered. A grey box means that a cabinet is unpowered, while a gradient indicates that the only parts of the cabinet are powered.

Finally, the context view shows the effect of the power management subsystem on the availability of the rest of the system. For example, if the system is turned on, the X-ray generators will receive power. However, the X-ray generator itself is not yet available for operation.

As a final note, the views presented here are the result of an iterative development process. These iterations included discussions on which elements to include in the views, the state of these elements and what kind of indications is useful. The views for each system state visualize the operational view. Together with the established views in the A3AO (Figure 5.17 or Appendix H.2), they represent the knowledge that needs to be communicated.

5.3.3 Supporting Communication with an Interactive System Overview

In the previous subsection, the relevant knowledge to communicate the subsystem's behavior was identified. This subsection considers how to support the communication of this knowledge. In this perspective, especially requirement 7 in Table 5.1: *"support exploration and evaluation of system behavior"* is relevant.

Regular A3 Architecture Overviews are for instance created using Microsoft Visio [Microsoft, 2010b] and printed on A3 paper. Microsoft Visio is used in this case study as well to create the initial A3 Architecture Overview as well as the definition of all the separate views for each system state, by copying and editing the basic views. However, to create a full implementation that allows interaction with these views and a coupling with the simulation model, further development is required. The simulation model, which was

already developed when starting the case study, has been implemented using the POOSL language [Putten and Voeten, 1997] in Eclipse [Eclipse, 2015].

The end goal of the implementation is to create an interactive system overview that supports communication of system behavior. To reach this end goal, three options are available for the implementation, based on the fact that Eclipse and Microsoft Visio are already used.

The first option is a full implementation in Microsoft Visio using Visual Basic for Application (VBA), building on the graphics already constructed in Visio, or using the layer function to hide and show different perspectives, as is done in [Kempen, 2015]. The main advantage of this is that the implementation is fairly accessible and portable. However, the existing simulation needs to be translated to VBA.

A second option is to export the graphics from Visio and import them in the Eclipse environment, making use of the graphics there. Exporting the graphics to Eclipse and linking them to a POOSL interface allows the view construction and initial validation to be done from Visio while a separate application for the communication can be developed. This preserves Visio's main strength, visualization, and Eclipse's main strength, the simulation and user interface construction.

The third option is a full Eclipse based simulation and the construction of the graphics completely in Eclipse as well. The risk is that the visualizations are harder to observe and validate as they are embedded in the code, but the advantage is that when more and more states are introduced, they are easier to maintain. Furthermore, it is easier to resize the UI if all graphics are generated by Eclipse.

Naturally, further options are available as well, but in this context it is logical to at least use either Eclipse or Microsoft Visio as part of the work is already done in these tools. In this case study, the second option is chosen. This is the most straightforward option to implement the interactive system overview. The final implementation consists of a Java [Oracle Corporation, 2015] based user interface constructed using Eclipse WindowBuilder [Eclipse, 2015], which communicates via a socket connection with a POOSL model. The Java user interface, socket connection and POOSL model were already established by [Schuts and Hooman, 2015a]. The extension analyses incoming messages over the socket connection and translates those to the required system state. For each system state, it is programmed which views are associated with the system state and they are shown on screen. The interactive system overview was distributed as a standalone application that can be used on the A3 tablet already discussed in section 5.2.3 and shown in Figure 5.15.

5.3.4 Using the Interactive System Overview

This section focuses on the actual usage of the interactive system overview. The interactive system overview was used during design discussions with stakeholders directly involved with the subsystem redesign (the researcher and two domain specialists) and during a design review of the subsystem behavior with additional stakeholders.

During the design discussions that were mostly held with three persons, the A3 tablet [Dell Corporation, 2015] was used to interact with the system overview and to explore the system through various states. The goal of this exploration was mainly to determine

whether the interactive system overview was working as intended, but in the meantime, the system behavior was further defined and formalized as well.

During the design review, six people were present, ranging from engineers to system architects. Observing an A3 sized tablet simultaneously with six persons is difficult. Therefore, the interactive system overview was shown on a big screen broadcasted directly from the tablet. Control of the system overview was done by passing around this tablet. Participants either controlled the overview themselves, or asked the person holding the A3 tablet to give a certain input. Both ways of working seemed to go well, however readability of text was an issue. An observation that was made during this design review was that participants were able to connect the different views. For example, one participant pointed out that if a certain component was receiving power in a particular system state that was being discussed, additional functions could be enabled in the system overview. The design discussion resorted to whether this behavior was desirable or not. From a physical perspective it seemed to be possible to switch on these functions without additional cost, as enough power was available. However, in the end the choice was made not to enable this functionality, to maintain a similar user experience with regard to functionality across several system states.

This subsection, as well as the previous subsections has explored the research goal of how to support the communication of system behavior. This was explained by illustrating how the support was implemented and used in a design case. The development of the support in this case study was a logical extension of the support discussed in the previous case study. Finally, it was shown that it is possible to use an A3AO interactively to discuss system behavior.

5.3.5 Discussion

Several research goals were addressed in this case study. The research goals were to identify the knowledge to be communicated and the support of this communication. A further evaluation of this case study is embedded in the general evaluation of the method, which is discussed in Chapter 8.

The first research goal was to identify the knowledge to be communicated. The approach used in this case study consisted of linking an existing simulation model with the overall knowledge that was available through the documentation and stakeholder expertise. By constructing an A3AO overview and identifying relevant system states, relevant views were identified. By essentially visualizing the A3AO for each system state, behavioral knowledge was captured.

To support communication of this behavioral knowledge, which was the second research goal, an interactive system overview was developed. The interactive system overview allows stakeholders to consider the system from various viewpoints. The contextual viewpoint proved to be very useful in connecting the power management subsystem to the rest of the system. Furthermore, various suggestions were given by stakeholders in the case study for the development and usage of the system overview. For example, participants in the design review indicated that the interactive overview was still quite technical and that views were

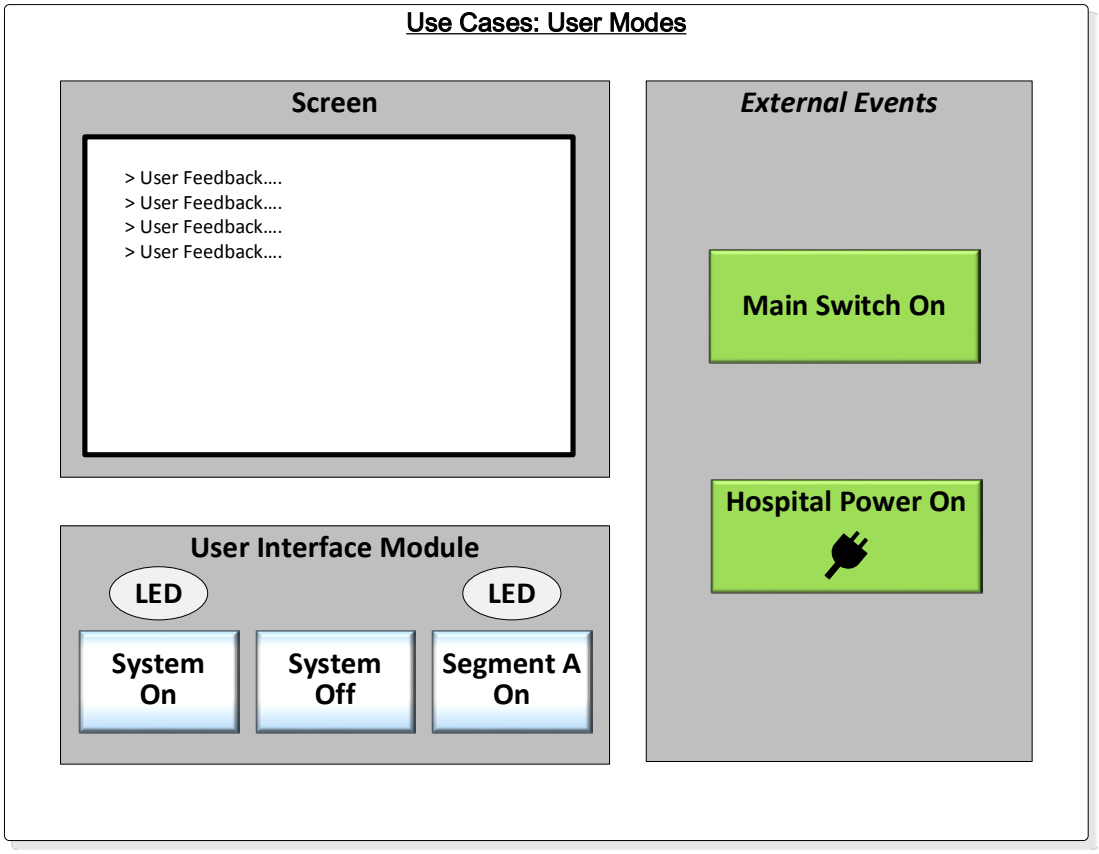


Figure 5.21 – User interface for the interactive system overview from the users perspective

lacking to discuss this with stakeholders from the marketing department who represent the user's perspective.

Furthermore, the current simulation does not represent the use cases from the physician's perspective. In order to address this, the simulation interface can be redesigned. An example redesign can be seen in Figure 5.21. In this user interface, the actual controls and feedback mechanisms of a physician are visualized. The controls are the "System On", "System Off" and "Segment A On" buttons, while feedback is provided in the form of indicator LEDs and the actions on the screen. "External events" were introduced to increase the number of use case that can be experimented with. For example, the physician is not involved with the mains switch (the system is always powered from a user perspective) and self-evidently does not control when hospital power will fail. By providing these "external events" some more behavioral aspects of the system can be explored.

Applications that not have been researched but that were suggested include amongst others the use of the overview to communicate with stakeholders from the manufacturing department. However, the overview was used to explain the system behavior to suppliers. This means that the overview is not only used in design discussions, but also as a tool to explain the system, both in structure as well as in behavior. This is similar to how A3 Architecture Overviews [Borches, 2010] can be applied.

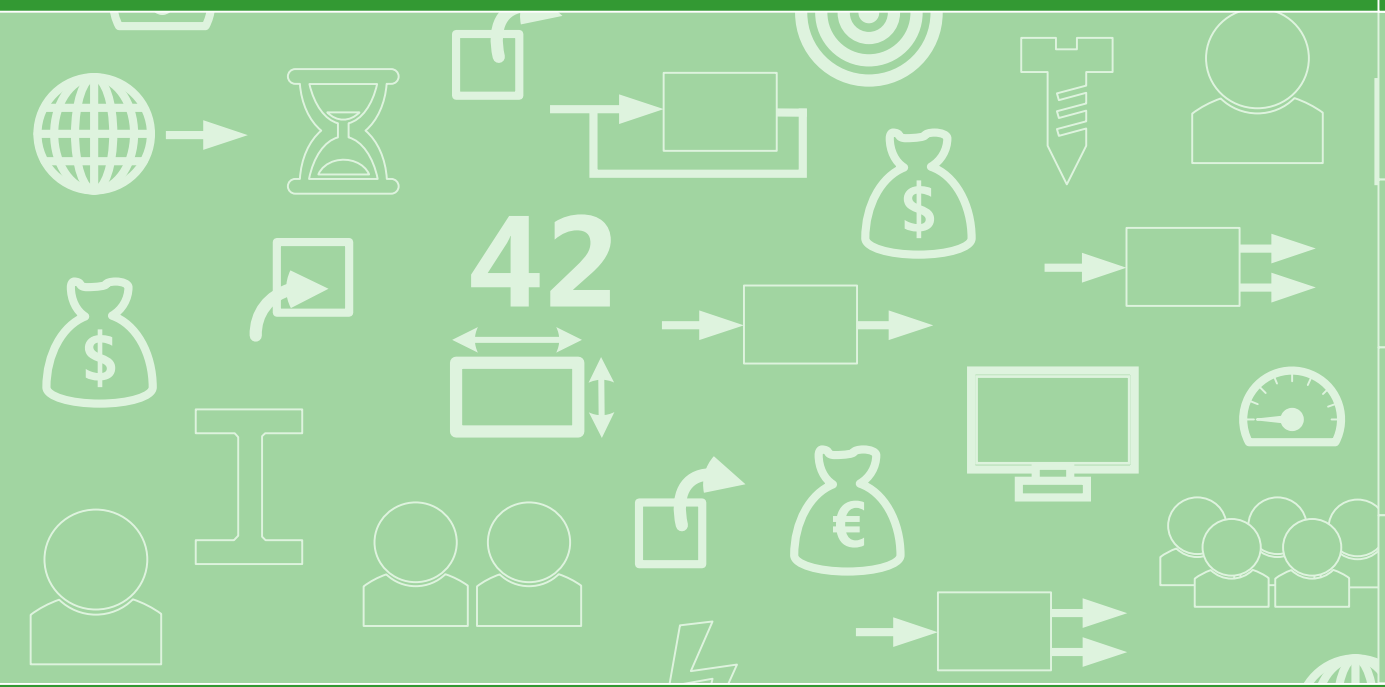
5.4 Conclusions

This chapter has described two case studies that both aimed at the communication of system behavior. The hand-eye coordination case study initially focused on uncovering behavioral aspects. During the case study it became apparent that an important and poorly supported aspect of doing these simulation studies is how to structure the communication of the model and its results. An A3 Architecture Overview was made to support communication. While this was partly successful, the case study also showed that simulation studies result in a multitude of system models that need to be explored. Therefore, some form of interactivity is required to present and encapsulate this data.

The second case study immediately focused on supporting communication of system behavior based on a simulation model of the start-up and shutdown behavior of the Philips interventional X-ray system. In this case, even though a model was already established, a system overview was lacking. Therefore, an initial system overview was created that helped to identify and summarize relevant knowledge. During the creation of this system overview, it became evident that it is possible to communicate the system behavior using this system overview, by detailing this system overview for each state, and by aggregating these in an interactive system overview. The experiences with this interactive system overview indicate that stakeholders are able to connect different views while discussing the system's behavior.

Finally, in section 5.1, requirements for the intended support were given. The actual support [Blessing and Chakrabarti, 2009]} presented in the first case study focused initially mainly on requirement 6 and 7 in Table 5.1. However, when using an interactive system overview as support it is possible to meet all requirements.

The COMBOS Method



1
2
3
4
5
6
7
8
9
10

6

The COMBOS Method

This chapter presents the COMBOS (COMmunicating Behavior of Systems) method, which aims to support communication of complex system behavior. The previous chapter described various activities performed within the case studies as well as the development of tool support. This chapter structures these activities by describing a method to perform and communicate a behavioral analysis during conceptual systems design. It also provides design guidelines that offer a foothold to execute the method. Design guidelines are rules, principles and heuristics that are useful to follow in attaining some design objectives [Blessing and Chakrabarti, 2009]. The guidelines will be mentioned throughout this chapter, where appropriate. An overview of the guidelines presented in this chapter can be found in Appendix D. The chapter also describes how to construct and implement a tool that supports this method. This tool is an interactive system overview.

First, this chapter addresses the scope of COMBOS in section 6.1. An overview of the method is given in section 6.2. Section 6.3 describes the method in more detail, while section 6.4 discusses the implementation of the method. Finally, section 6.5 presents the conclusions to this chapter.

6.1 Scope

While the aim of COMBOS is to be applicable on a wide range of conceptual system designs, the scope has to be reduced somewhat to ensure this applicability. This section highlights several of the choices made in this regard.

6.1.1 System Types

As stated in section 2.6, the research in this project mainly focuses on single-entity product systems [Blanchard and Fabrycky, 2010]. These kinds of complex engineering systems are often OEM-type products. An important emphasis is that while the system itself may be considered as a single entity, the broader context in which the system operates is considered as well. Out of scope are biological and societal systems, systems that exclusively deal in a single domain (such as software) and Systems of Systems, though the method may be useful in these areas as well. This is further discussed in section 8.3.

6.1.2 Project Types

With the system scope defined, it is now also possible to define the types of projects, design processes and design questions that the method is applicable to. In section 3.1.2, four different types of conceptual system design processes were identified. The method discussed in this section is useful for three of these four types, being new system design, system redesign and subsystem redesign. Design processes of the fourth type, subsys-

tem/component update, will be able to test behavior very quickly on the actual system. This reduces the need for simulation studies. The following guideline reflects that notion:

Guideline 1; *exploring system behavior (using this method) is mainly useful if the conceptual design process does not allow a "quick and dirty" way to test the new system via a prototype*

6.1.3 Users

The main actor within this method is assumed to be the system architect. However, the method explicitly aims at involvement of other stakeholders. This involvement is essential to the success of this method. The behavioral analysis will have to be executed by the system architect or the system architect needs to work closely together with a simulation engineer. As a system architect often lacks the time and resources to conduct a full in-depth analysis on a single issue, it is assumed that often a simulation engineer collaborates with the system architect. With regards to the skills required, [Morse *et al.*, 2010] state that good conceptual modelers are computer scientists with domain expertise. If either skill is lacking, the conceptual modeler has difficulty building a model that bridges the gap between the real world and the computation space. The inclusion of a simulation engineer introduces an important communication channel between the simulation engineer and system architect. Therefore, this can be characterized as a more complex process than the system architect working alone.

Ultimately, the owner of the created views and the knowledge contained within them is the system architect. The owner of the supporting model and the technical infrastructure can either be the system architect or the supporting simulation specialist. Finally, the identified stakeholders that are relevant for the system are closely involved in the development of the overview as well. This leads to the following guideline:

Guideline 2; *while a simulation engineer might be required to create an interactive system overview, the system architect is responsible for deciding which viewpoints are used to construct views in the interactive system overview*

6.2 Overview

The main goal of COMBOS is to support communication of behavioral systems aspects. In order to achieve this, the method aims to communicate behavioral analysis efforts by constructing and using an interactive system overview that contains the views that have been defined in Chapter 3. Every interactive system overview captures a single concern, although other concerns are certainly represented in these overviews.

The case studies described in Chapter 5 and especially the case study described in section 5.3 showed that in order to communicate the operational view, it must be represented throughout the other views that describe the system. This aligns with the 4+1 framework of [Kruchten, 1995], who advocates that the "+1 view", the use case view, should be represented throughout the other views. This definition is the underlying principle for COMBOS. This principle is to couple the operational view with the functional, physical and quantification view in an interactive system overview. This coupling is visualized in Figure 6.1. If this

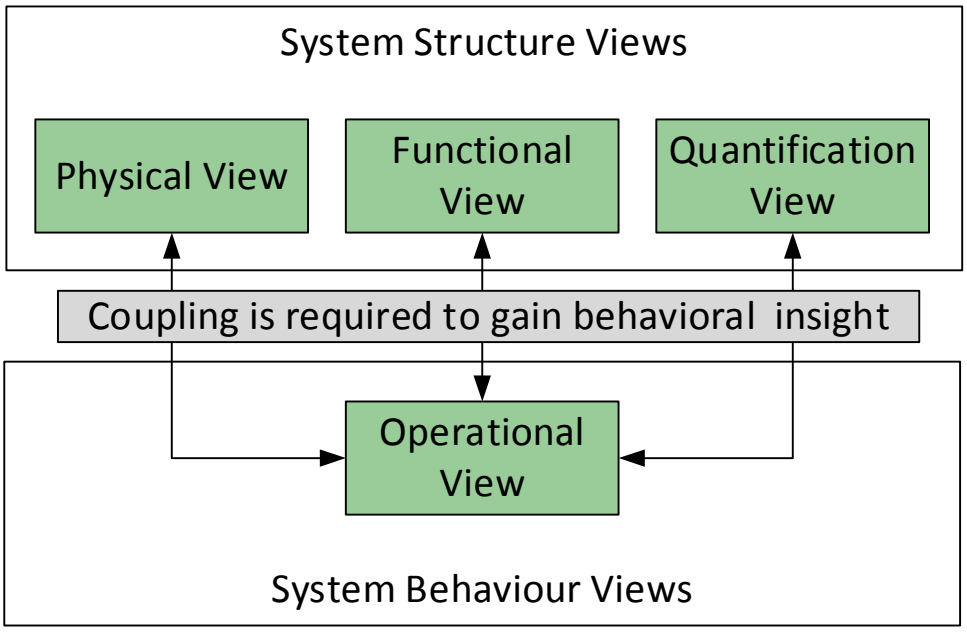


Figure 6.1 – Coupling between views that describe system structure and system behavior

coupling is not made explicit, a stakeholder has to translate a described scenario or use case to his own view. This requires significant mental effort and is prone to misinterpretation. Furthermore, this figure also shows that the physical view, functional view, and quantification view represent the system structure, while the operational view represents the system behavior. This can also be summarized in the following guideline:

Guideline 3; *to represent and communicate the operational view, it must be coupled to other views, such as the physical, functional or quantification view*

COMBOS aims to support three types of knowledge creation according to the definition of [Nonaka and Takeuchi, 1995] (see Table 2.2 for an overview).

First, *externalization* of knowledge on the particular aspect that is being analyzed, using the approach of the A3 Architecture Overview method [Borches, 2010]. This will allow stakeholders to understand the system from their viewpoint.

Second, *combination* of various views that represent the same behavior from different viewpoints allow stakeholders to discuss the system across discipline boundaries as there is a clear association between these views [Alvarez Cabrera, 2011] (see also section 2.3.2). These discussions can lead to knowledge creation, as for example was shown in the case study in section 5.3.4 where stakeholders were able to identify new possibilities for the system by combining views.

Third, *internalization* of knowledge is also supported by the method, as a stakeholder can observe the system behavior presented in the interactive system overview on an individual basis.

Finally, *socialization* is not an explicit aim of this method. This type of knowledge creation is addressed in collaborative design approaches as for example at the ESA concurrent design facility [ESA, 2015]. However, COMBOS can be applied within this context.

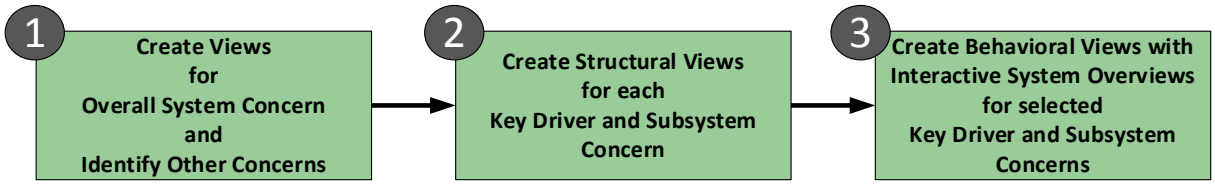


Figure 6.2 – High level process description for the COMBOS method. A detailed stepwise description is available in Appendix D

The interactive system overview that results from applying the method allows the user to manipulate the system and its context. The effects of these inputs are shown in the physical, functional and quantification view. This allows stakeholders from multiple disciplines to understand and assess the system in different operational modes. The differences between these operational modes will support understanding of the system behavior. Design discussions can for example focus on whether changes occurring due to a specific input are desired, whether a system behaves properly in a certain state or what the influence of certain inputs is on the resulting system behavior. This leads to the following guideline:

Guideline 4; *visualizing the operational view simultaneously in different views allows reasoning on system behavior from different perspectives, which in turn enables multidisciplinary design discussions, and ultimately, new knowledge creation*

In order to establish this system overview and ultimately enable communication of behavioral system aspects, the method employs a three staged process shown in Figure 6.2. Step 1 and 2 of this process mainly focus on establishing a system overview as part of the conceptual system design process. The actual communication of system behavior is mainly contained within step 3.

6.3 Detailed Description

This section describes the method overview presented in Figure 6.2 in more detail. Each step is treated in its own subsection. As the third step includes the largest contribution of this research, this step is discussed in more detail than the other two steps.

6.3.1 Step 1: Characterizing the Overall System

First, the system is defined at the highest level, by constructing views for the overall system concern and identifying the other concerns. This can be realized as described in section 3.3.2, by employing a combination of systemigrams [Boardman and Sauser, 2008], FunKey diagrams [Bonnema, 2008] and storytelling to define use cases [Muller, 2004]. Ultimately, this leads to a definition of the key driver and subsystem concerns using the view presented in Figure 3.4.

The context view, which can be visualized with a systemigram, also includes the identification of stakeholders. The fact that stakeholders are included in the systemigram implies that they are affected by the system. In this stage, it is important to already involve those stakeholders in the system design process. These stakeholders can be involved directly but

also through an intermediary. For example, within Philips iXR, application specialists and clinical scientists act as an intermediary between the design team and the hospital staff.

[Grogan, 2014] states that in practice, effective learning from models occurs best and perhaps only, when the decision makers participate actively in the development of the model. Modeling here includes the elicitation of the participants' existing mental models.

One of the main pitfalls in communicating simulations is to only communicate at the start and end of the analysis process [Kahn and Mann, 1957]. The final results do not give insight in the behavior of the system, but the knowledge gained during the creation process of the simulation. This is also why many modelers claim to understand a system appropriately after modeling it, as they have gained the knowledge while creating the end result. However the process followed to arrive at this end result is oftentimes hard to understand for an outsider. Therefore, it is important to continuously involve relevant stakeholders in the simulation process [Morse *et al.*, 2010, Topper and Horner, 2013]. This is summarized in the following guideline:

Guideline 5; *ensure continuous communication throughout the process with affected stakeholders, especially the decision makers, focusing on construction and validation of the created views in each step*

In summary, the first step comprises the creation of various views. These are a functional view, the operational view and the context view. During the creation of the context view, the stakeholders are also identified. Involvement of stakeholders in the remainder of the process when constructing and validating the views that are created is important.

6.3.2 Step 2: Creation of Structural Views

The second step of COMBOS focuses on detailing the concerns using structural views. These concerns are elaborated with structural views (see Figure 6.1). The core of the structural views is the functional, physical and quantification views. However, each of these views can be represented in different ways varying in content and viewpoint. Also, depending on the issue at hand, other views than the functional, physical and quantification view can be relevant as well, or one of these three views can be emphasized less.

In general, the approach as described by [Borches, 2010] to create A3 Architecture Overviews should be followed, especially the guidelines for creation of the model side of A3 Architecture Overviews [Borches, 2010, p114]. The actual construction of views depends on the specific case. Therefore, new developments should draw upon the examples given in this thesis, both in Chapter 5 as well as the examples that will follow in Chapter 7.

In the system latency case described in section 5.2, the functional, physical and quantification view were encapsulated into one single view. The quantification view in this combined view focused on quantification of the system model elements. Central to the communication of this particular system issue was an operational view coupled to a quantification view of performance parameters. Furthermore, a number of visual aids explaining system concepts were present.

In the power distribution case described in section 5.3, the quantification view was used as a reference in the system overview, and was not used to show changes in parameters

during a system's operation. The behavior was only represented in the functional and physical view. However, an extra view, the context view was added in which the influence of the subsystem on the overall system and the stakeholders in its context were made visible.

Visual aids support the comprehension of the views for stakeholders. The introduction of visual aids follows naturally when creating the views. For example, when validating a view with a stakeholder, this stakeholder might not understand a certain concept. Additional information in the form of a drawing, text or a table is provided. The logical consequence is that this additional information can be utilized as a visual aid.

In conclusion, the second step concerns the creation of structural views for each of the concerns, either from a key driver perspective or a subsystem perspective. Not all concerns need to be elaborated with structural views, as only the structural views that will be extended with behavioral views need to be established. However, to obtain a good overview on the system architecture in the conceptual system design phase, it is highly recommended to ensure views are created for all concerns. This is summarized in the following guideline:

Guideline 6; *the structural views for the concerns that are elaborated with behavioral views must be established. It is also highly recommended to create structural views for all concerns, as this ensures that the full system architecture is considered*

6.3.3 Step 3: Creation of Behavioral Views

This subsection discusses the third stage of COMBOS as described in Figure 6.2. In this stage structural views are extended to also include the behavioral views. For a fairly simple system, system behavior might be directly inferred from the structural system views. However, for most concerns, a substantial amount of behavioral analysis supporting the conceptual system design is required. Therefore, the goal of the method in this stage is threefold.

First, to identif y whether and what kind of analysis is required. Stakeholders must be supplied with information that provides a sufficient foundation to make design decisions. They must also gain the confidence that additional analysis does not further improve the conceptual system design. Therefore this step is both the opening as well as the closure for the behavioral analysis, because it indicates whether to either start, continue or stop the process. Second, is to support the creation of a behavioral analysis, and third, to be able to represent this analysis in such a way that involved stakeholders understand the analysis and

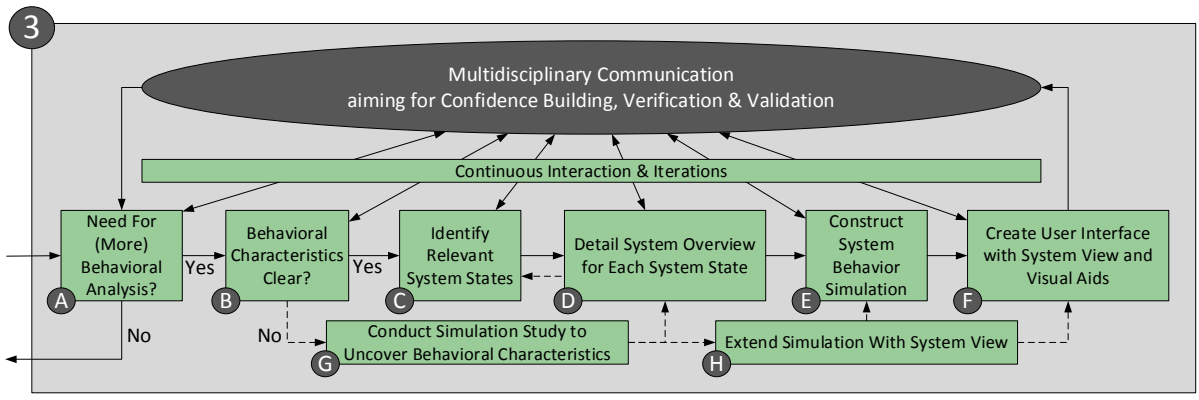


Figure 6.3 – Detailed approach for third stage "constructing behavioral views"

its implications for the total system under design. Figure 6.3 details the process of the method for the third stage in Figure 6.2. The following paragraphs will explain the steps in this stage in further detail.

Step 3A – Identifying the Need for Behavioral Analysis

This step concerns identification of the need for behavioral analysis and the definition of the analysis goals. It also includes the decision making process of whether the analysis that has been performed is sufficient. This section discusses three identified situations where behavioral analysis is highly beneficial. These are based on case study experiences as well as the types of complexity identified by [Suh, 2005]. Once a need for behavioral analysis is identified, goals should be established for the simulation process. This is summarized in the following guideline:

Guideline 7; *once a need for behavioral analysis is identified, translate this need in well-defined goals for the analysis process. These goals should act as a reference during creation of the interactive system overviews*

This section further discusses the issue of how to decide whether an analysis is completed. The first situation in which behavioral analysis is beneficial occurs if the system has a high number of relevant states, as for example in the power distribution case in Chapter 5. A lot of requirements that include a statement on the state of the system are a clear signal for this. A requirement or need such as "the system must ensure that all captured X-ray image data is stored, even if normal operations are disrupted" is a good example of this. In this case, a behavioral analysis can help to determine whether the system behavior corresponds to what was intended for a particular state. [Suh, 2005] refers to this as time-independent real complexity, as interactions either in the system or within its context might interfere with the intended system behavior.

The second situation occurs if there is variance in internal parameters of the system, as for example in the system latency case in Chapter 5. If dependencies between different parts of the system are low, simply summing the range of the parameters for the different elements will give a good indication of the range for the whole system. However, if dependencies are present, this does not work. For example, it could occur that an image is handled by component A, but component B is still dealing with the previous image, and is thus unavailable to process the next image. In these cases a simple summation does not work anymore and a behavioral analysis is required. According to [Suh, 2005], this is time-dependent complexity, which can either be unpredictable (combinatorial) or predictable (periodic).

As opposed to the second situation, the third situation is that the number or type of inputs generated by the context can vary greatly. This is variation that is external to the system. An example would be that depending on the mission trajectory of a spaceship, the risk of colliding with asteroids or space debris has to be analyzed. These outside influences possibly require evasive maneuvers and need to be accounted for. A behavioral analysis shows how a system behaves in these situations and what consequences this behavior has.

In the example, this could be additional fuel usage for evasive maneuvers. This is also time-dependent complexity according to [Suh, 2005].

[Suh, 2005] identifies a fourth type of complexity that was not treated yet. This is time-independent imaginary complexity, which implies that designers are ignorant of the existence of this complexity. Therefore it is hard to identify a situation where this can act as a trigger for behavioral analysis. However, this kind of complexity can be uncovered through behavioral analysis as in the case study discussed in section 5.2.1 it was found that the threshold for the glitch definition could be enlarged.

The second part of this step concerns identifying whether a behavioral analysis is necessary or when it is completed. Additional analyses or further detailing an analysis are large temptations. From a theoretical perspective, the value of information theory (as discussed in section 2.4.1) provides a metric for determining whether it is valuable to start or continue an analysis. [Thompson and Paredis, 2010] discuss an approach to quantitatively determine whether or not an analysis process should be performed. This relies on quantifying three aspects: (i) the expected cost of performing the analysis, (ii) the amount of uncertainty that an alternative to be explored contains and (iii) the expected reduction of uncertainty through the analysis effort. The method that is presented in this chapter is not based on quantifying these kinds of numbers. It has also been experienced that the amount of assumptions that have to be made before reaching a decision in this way would slow down the decision process [Balk, 2015]. Therefore, it is infeasible to utilize this approach within the COMBOS method.

During the case studies it was observed that initially, stakeholders asked a lot of "what-if questions" that indicated a need to explore the behavior of the system concern under analysis. Once the outcomes of the simulations behaved in line with expectations of stakeholders, these questions were asked less and less. In the case studies, this was the point where the analysis efforts were stopped. While an exact indication cannot be given, a consensus of stakeholders on the observed system behavior and a reduced need to explore the system aspect indicate that a behavioral analysis is finished. This leads to the following guideline:

Guideline 8; *keep track of the questions that stakeholders have with respect to the system behavior. If the number of questions stabilizes, either due to the fact that they are answered or cannot be answered, stop the analysis or transition to the next step*

In summary, the trigger for behavioral analysis stems from the need to either look closely at system states, to investigate influence of parameter variation or to determine the influence of a changing system environment. Of course combinations between these types are possible. Furthermore, while it is possible to quantify the value of continuing with an analysis, using a heuristic for this decision is more applicable for this method.

Step 3B – Behavioral Characteristics

Based on the case studies discussed in Chapter 5, it is identified that two main ways of conducting a behavioral analysis are possible. The first approach concerns a step-by-step analysis extending a static model as is applied in section 5.3, while the second approach

aims to define the behavioral characteristics directly through a simulation model. This approach is followed in section 5.2.

Whether to choose this first approach, which follows steps 3C to 3F, or the second approach, which follows steps 3G & 3H depends on whether the behavioral characteristics are clear upfront. In the hand-eye-coordination case study discussed in section 5.2, it was unclear which factors determined the behavior upfront. These were only uncovered through a simulation. This made it impossible to directly start with construction of the interactive system overview, as the relevant views were still unclear. In the power management case study, all possible inputs that affect the system were clear upfront, therefore, it was possible to directly start with construction of the views necessary for the interactive system overview.

Step 3C to 3F: Detailing System Behavior

If the behavioral characteristics are clear, the analysis can start with identifying the relevant system states directly. The goal for these steps is to visualize the system in each system state, by detailing the system's functional, physical and quantification view for those states. Stakeholders can discuss whether or not certain properties of the system in a particular state are desired. However, not only the states themselves are subject of analysis, but also the transition between states. In this case, formalizing the states and transitions in a simulation model allows users to control, verify and validate the system behavior with respect to stakeholder expectations and posed requirements. Such a rule-based simulation checking for transition conditions helps to uncover how a system reacts to certain inputs. Eventually, these types of models also allow model based verification of a set of requirements [Hooman *et al.*, 2012]. This can be summarized in the following guideline:

Guideline 9; *when considering the behavior of a system across states, both the behavior of a system in that state, as well as the possible transitions to other states should be subject of interest. Simulation models explicitly allow exploration of these transitions*

Therefore, steps 3C to 3F comprise the following activities. Step 3C identifies the system states that are relevant for the issue at hand. The system is detailed in Step 3D for each of these states in the functional, physical and quantification view. The next step, 3E, is to formalize these states and transitions in a simulation model. The final step is 3F, which aims to create a user interface to connect the various system states and allow users to move from one state to another. The simplest example would be to simply visualize each system state for all the views, put them in a slide presentation and go through them one by one. However, to connect better to end users and customers, it is very valuable to provide a user interface simulating the available system controls and certain outside input (such as a mains power failure in the power distribution case described in section 5.3).

Step 3G & 3H: Uncovering Behavioral Characteristics

If the behavior of a system is less clear upfront, or the reasons of why an existing system behaves as it does are unclear, this needs to be analyzed first. In this case, a simulation study should be used as a starting point for the behavioral analysis. Thus, the initial focus of the analysis is oriented on characterizing the system behavior and finding a good model

to represent this behavior. The creation of this simulation should follow the process described in section 4.1.3. Following this process assures that an appropriate simulation for early, conceptual system design is created. Step 3G can also be taken as a starting point if for example a simulation is already present in the organization, as occurred in the case study described in section 5.2. One should however pay attention to whether this existing simulation appropriately focuses on the conceptual system design. This can be done by using the framework described in section 4.2.6.

After these steps are completed, the goal is to extend the user interface and visualization of the simulation to include the essential views that show the big picture of the system under design. To do this, steps 3C to 3F can be followed.

Final considerations for step 3

In conclusion, continuing with step 3G after step 3B is appropriate when the behavioral analysis focuses on discovering key characteristics of the system behavior that influence the system. This could be seen in the system latency case presented in Chapter 5 as well. Generally speaking, this approach is useful when analyzing system behavior for key drivers. The other approach, continuing with step 3C after 3B is the approach that should be used when behavioral characteristics are clear and the focus is on detailing the system behavior across the systems states. In general, this approach seems more useful for subsystem concerns. As a final note, it is of course possible to combine the two approaches if a particular concern requires that approach.

6.4 Implementation

While the case studies in Chapter 5 already gave an indication of how the method can be realized, this section describes the implementation of the method. Based on the fact that conceptual system design is a fairly informal process, highly formalized tooling is not a prerequisite. In fact, there is an inherent opposition between understandability and formality [Bonnema, 2014]. This supports the conclusion that for example the formal approach used in Model Based Systems Engineering is less suited for communication, and is more suited to ensure re-use, consistency and integration [Woestenenk, 2014].

An overview of the areas that will be discussed in this section is given in Figure 6.4.

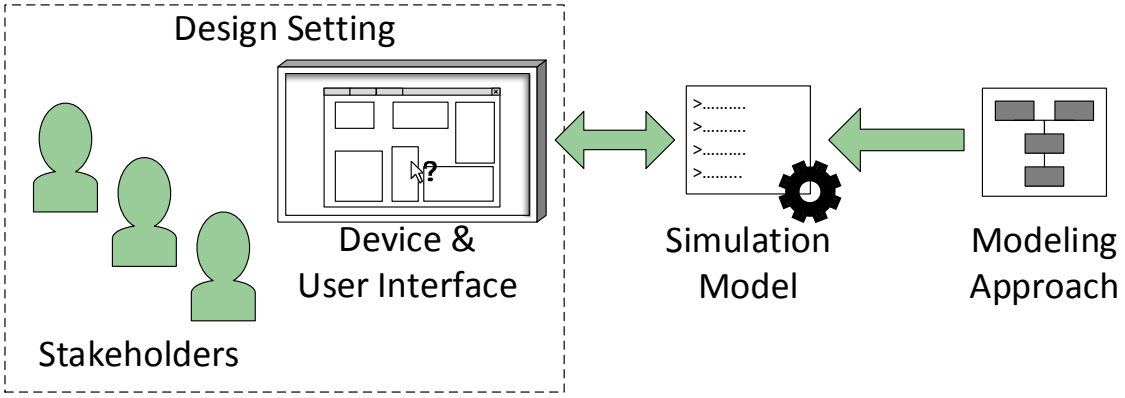


Figure 6.4 – Overview of areas relevant for implementation of COMBOS method

These areas are the device and user interface that make up the communication medium for the interactive system overview (section 6.4.1), the underlying simulation model and modeling approach (section 6.4.2) and finally the setting in which the method and interactive system overview is used (section 6.4.3).

6.4.1 Communication Medium

In order to be able to use and share interactive system overviews, a communication medium is required. This communication medium is composed of a user interface and a device on which the user interface is presented. Central to the communication is the representation of the systems structure and its behavior. As the paper based A3AO format limits insight in scenarios and consequences of changing key system concepts, COMBOS employs the use of digital, interactive system overviews.

In order to preserve the overview that a paper based A3 Architecture Overview is based upon, the device that is used should provide a screen size similar to an actual A3 paper. At the same time, the user interface provides a way to encapsulate and present changes to the system over time and thus allows discussion of the system behavior. The device requires a large screen size because it allows multiple views to be shown at the same time. A user is able to observe and link these views as they are presented together. If the screen size is reduced, this is impossible anymore, as either readability or the number of views that can be shown decreases. If the screen size were to be increased to show even more data, there is too little abstraction and users can easily be overwhelmed. The use of a tablet like the Dell XPS 18 [Dell Corporation, 2015] that was utilized in the case studies allows a similar use setting to a paper based A3AO, as it can be placed on a table and passed around. The specific guideline for this is formulated as follows:

Guideline 10; *to communicate an interactive system overview, preserve the size of a paper based A3 Architecture Overview (A3 size). Therefore, aim to use (portable) devices that have a diagonal screen size of 18-22 inch and support resolutions of 1920x1080 or higher*

By digitalizing A3 Architecture Overviews, a great number of options for interaction with the overview are possible. [Heer and Shneiderman, 2012] group these interactions in three categories, which are data & view specification, view manipulation and process & provenance. Data & view specification includes visualization of data by choosing visual encodings and filtering out data. These tasks were described in section 6.3.2. View manipulation is the category that mainly deals with the content of the interactive system overview, by selecting information, navigating and organizing. Process & provenance supports dealing within interactive system overviews by recording information, annotation and guiding users through the overview. Several useful interactions were described by [Brussel and Bonnema, 2015]. The focus of their research was to explore the possibilities of interactive A3AO's, while the research in thesis focused on utilizing interactivity to leverage knowledge on system behavior.

Table 6.1 gives an overview of useful interactions that can be used to populate the interactive system overview. Not all interactions have to be added to each interactive system overview. The added value of the interactions should be considered on a case by case basis.

Table 6.1 – Overview of possible interactions for Interactive System Overview

Interaction	Example	Goal
Adjusting Values	Changing values by entering a new number or choosing from a number of predefined options	Explore influence of parameters
Internal Links (Color)	Highlight active elements by using a green color and use a red color for an element that is unavailable. Highlight relevant design constraints in the design constraints list for a specific stakeholder	Communicate system status and make relations explicit
Internal Links (Arrow or Highlight)	Link the stars for design constraints with those placed in the interactive system overview with a line when clicked. Link two related concepts in different views with a line when clicked and show with an icon, highlight or color that a link is present	Make relations explicit
Zooming	Zooming on block diagram reveals lower levels [Brussel and Bonnema, 2015]	Encapsulation of information
External Link	Link to other Interactive System Overviews, further analysis details, design documentation, etc. [Singh and Muller, 2013]	Expand access to Knowledge
Manipulation of 3D Models	Allow users to manipulate 3D models [Brussel and Bonnema, 2015]. Showing effect of behavior in 3D Model [Hooman *et al.*, 2012]	Spatial representation of physical view
Audiovisual Explanation	Talk a stakeholder through the interactive system overview by recording an audio message, as a tutorial [Brussel and Bonnema, 2015]	Internalizing knowledge [Nonaka and Takeuchi, 1995]
Comments or Annotations	Adding comments and drawings or even directly adapting the interactive system overview [Brussel and Bonnema, 2015]	Reviewing and updating the overview

For example, the use of external links can be beneficial to expand the access to knowledge. However, it can also be highly confusing for users, as when they "leave" the interactive system overview, it can no longer act as a frame of reference.

In conclusion, the communication medium used in this method is an interactive system overview that makes use of a selection of the interactions presented in Table 6.1 and is presented on devices that have a screen size that is similar to A3 papers to preserve the overview and constraints imposed by that format.

6.4.2 Modeling and Simulation Approach

To be able to embed knowledge into an interactive system overview, the use of a modeling and simulation approach is required. As the communication medium used is based on the informal A3AO method, it does not make sense to use a rigorous modeling language like SysML specifically for the purpose of this method. It only has merit to use such a modeling language in the case that an overarching Model Based Systems Engineering framework exists in the organization where the method is applied. This possible connection is not part of this method. However, suggestions on how this can be achieved are given in section 8.3.4.

The choice for the Y-chart methodology is based on the fact that the Y-chart methodology uses the triad of functional, physical and quantification views, which supports multidisciplinary communication [Bonnema, 2014] and is especially suited for high level modeling (see also section 5.2.1). Supporting high level modeling is important, because it allows an exploration of the problem domain, instead of focusing on details in the solution domain. This in turn supports a well-founded conceptual system design process and is summarized in the following guideline.

Guideline 11; *when establishing simulation models in conceptual system design, aim to model only generic behavior and avoid going into too much detail as this distracts from exploration of the problem domain*

Discrete event simulation is generally suited for early design, as it focuses on functional reasoning. [Schuts and Hooman, 2015a, b] discuss discrete event simulation with the POOSL language in more detail and conclude that this type of simulation, applied as a lightweight method, is helpful in quickly formalizing the conceptual system design. It forced design discussions on issues that would have otherwise been postponed to detailed design stages. Therefore, the choice was made to use POOSL as simulation language in this method over other discrete event approaches such as ASD [Broadfoot, 2005] or Colored Petri Nets [Wang and Dagli, 2011] (see also section 5.2.2). During the research in this thesis POOSL has been implemented within the Eclipse Modeling Framework [Eclipse, 2015]. As the Eclipse Modeling Framework also allows Java graphical user interface development, using Eclipse allows the development of both the simulation model and user interface.

In conclusion, to create the models, including the simulation, the method utilizes the Y-chart modeling approach combined with the discrete event language POOSL. The development of the POOSL models and user interface can be done using the Eclipse Modeling Framework.

6.4.3 Design Setting

In Figure 6.2, the process for the COMBOS method is presented. The question is how this process is supported. This support can be embodied in various ways. For example, a complete tooling environment can be created that asks users step by step to fill in information. However, the broad scope that is employed in the conceptual system design does not fit a template format. Therefore, only a stepwise process description and guidelines are offered as support to execute the process. These are presented in Appendix D. This subsection focuses on how the method including the interactive system overviews can be used in a conceptual system design process. This usage takes place in different design settings, which can be for example small or large face-to-face meetings or through digital communication mediums.

During the execution of the method, day-to-day collaboration or less formal meetings is often done in small groups. In these settings, it is possible for the persons involved to discuss and control the interactive system overview together. Any of the persons is able to

easily point at specific elements in the overview or control the interactive elements of the overview. If more people are involved, this capability is lost.

With larger group sizes, it is less feasible to interact around a single overview when compared to smaller meeting sizes. Therefore, typically only one person is able to control the overview. With paper based A3 Architecture Overviews, it is possible to hand out an A3 to each person. Also, it is roughly possible to point out things on the A3 and have other people focus on that as well. Digital media can for example be presented via a beamer on screen. However, a high beamer resolution and large projecting area or a large video screen are required to show the complete A3 overview. In the case study in section 5.3, the interactive A3 Architecture Overview was used on a large TV screen, remotely operated from the A3 tablet. This allowed the control of the overview to be passed around, while maintaining the same overview for all persons present. Especially in this situation care has to be taken that the overview is still readable from a distance and that the person controlling the overview explains his actions.

The third setting is communication through online media. This type of communication lacks the ability to easily establish a common ground [Weck *et al.*, 2007]. Synchronous online communication can be supported with audio and video and for example with screen sharing. This can be especially relevant if stakeholders are located far apart. There have been no specific experiences in this research regarding synchronous online communication. Asynchronous online communication includes for example document sharing via e-mail. This form of communication has been used to send out files for review. In this case, the source files of the interactive overviews were distributed, so that recipients had a full list of all system states available for review. This way, all states are considered separately. Additionally, it is possible to distribute the complete interactive system overview for personal usage or to once again use it in small or larger group settings.

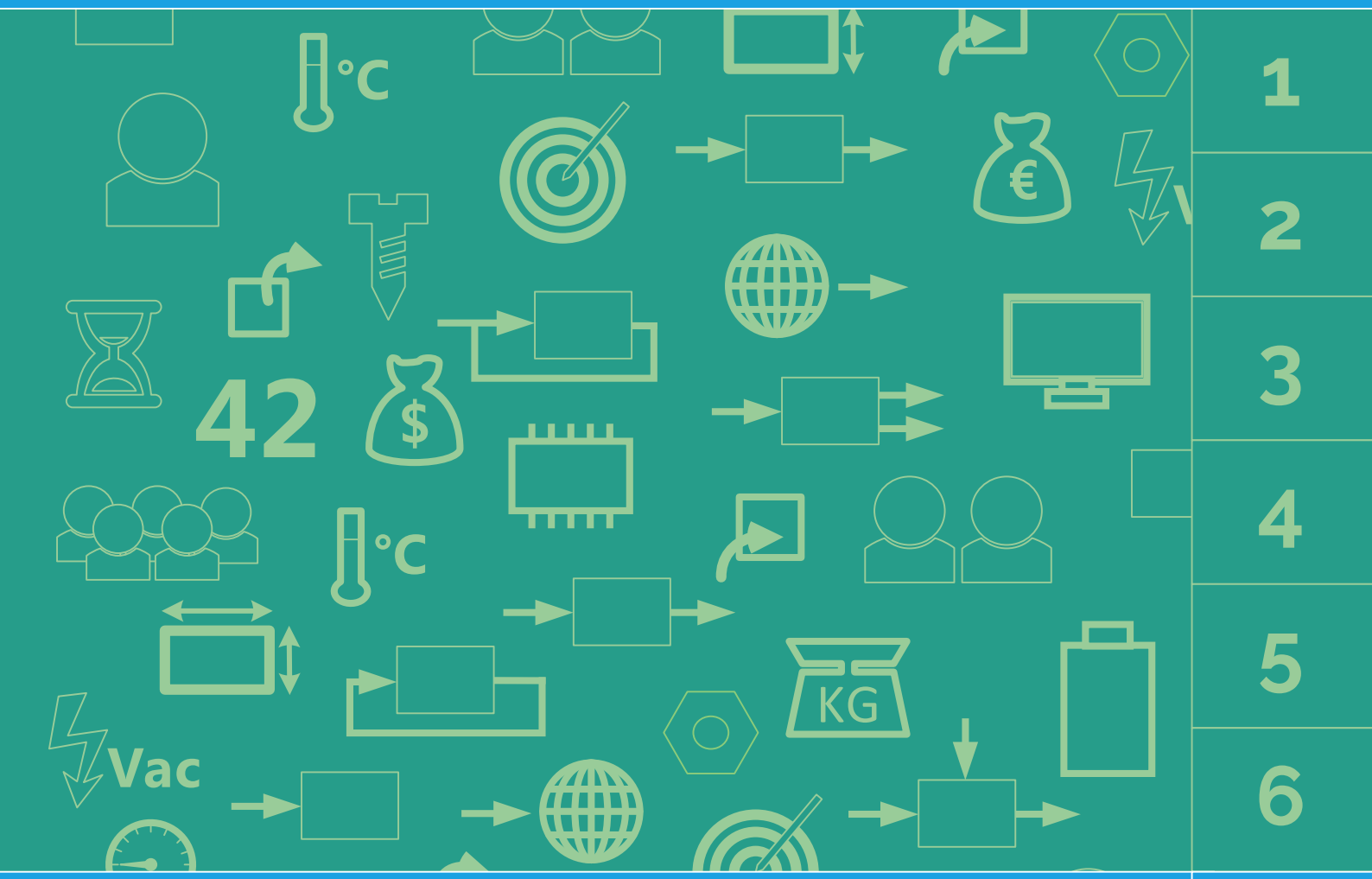
In conclusion, the usage of the interactive system overviews is most effective in small group sizes that are common in day-to-day collaborations. Larger group sizes make it complicated for all users to interact with the system overview, however, approaches to mitigate this can be devised. Finally, when communicating through online media either the complete tool or separate source files can be shared to do for example reviews.

6.5 Conclusions

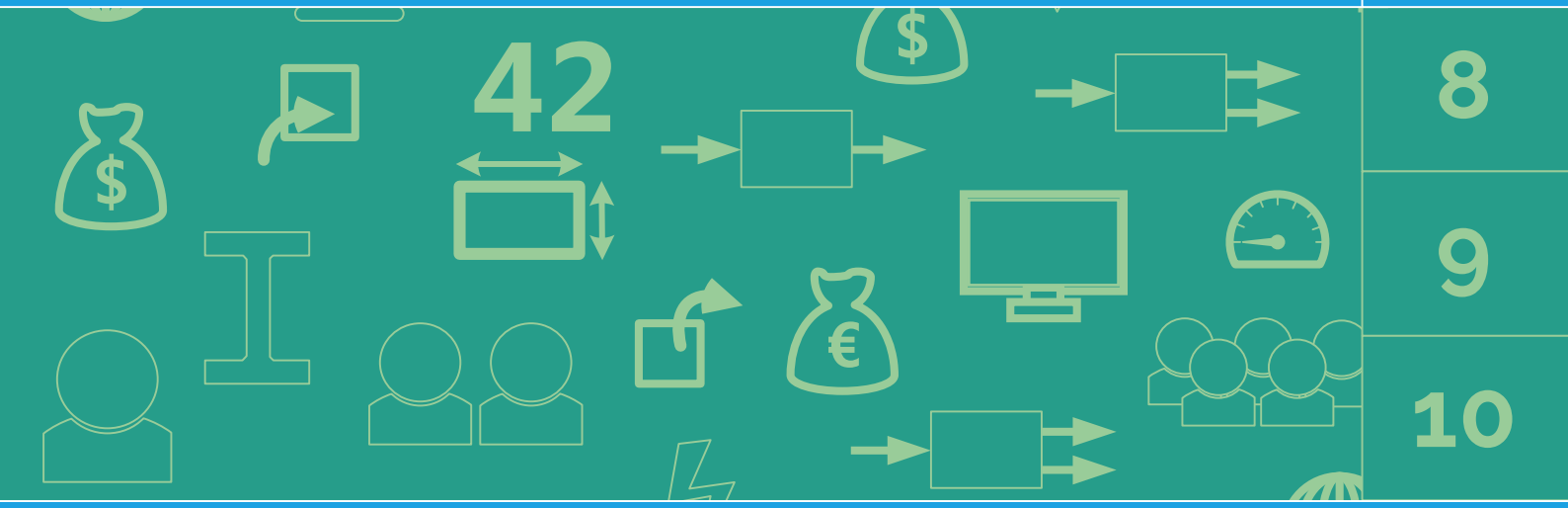
This chapter has presented the COMBOS method to communicate system behavior in conceptual system design. It does so by providing a step-wise process with continuous stakeholder communication that ultimately results in the creation and usage of an interactive system overview. One of the main goals of this interactive overview is to enable multidisciplinary design discussions through shared views on the system. Of course, some views are more useful for certain stakeholders. For example in the power distribution case (Chapter 5), stakeholders from marketing (representing the customer and user perspective) did not relate to the physical system view which showed which parts of the system were powered. In the same sense, hardware designers did not connect to the user mode views. However, by combining these views into one single overview, the relation between these

views become apparent and allows for mutual understanding, as also [Alvarez Cabrera, 2011] emphasizes (see section 2.3.2). This mutual understanding does not only come from design discussions based on the overview presented, but also arises throughout the process, because the process forces stakeholders to define the system in their view for a certain state. As this state is directly linked to other views as well, the whole system definition is improved.

To support execution of the method, a number of guidelines have been elicited throughout this chapter. They have been summarized in Appendix D. Appendix D also includes a detailed stepwise process description.



Examples of COMBOS Application



7

Examples of COMBOS Application

This chapter gives two examples to illustrate how the COMBOS method presented in Chapter 6 can be applied. Furthermore, the examples serve as an initial evaluation of the usefulness and applicability [Blessing and Chakrabarti, 2009, p195] of the method, as was also outlined in the research approach in section 1.5. When applying the method, continuous iterations with stakeholders are important. In this case, as this is a theoretical example, fewer iterations were used and choices in this regard were made by the researcher.

The first example in section 7.1 considers an electric vehicle and aims at characterizing system behavior across the different states of the system, similar to the case study discussed in section 5.2. The second example in section 7.2 focuses on identifying the main influences on the system behavior of a space mission that aims to retrieve a ground sample from one of the moons of Mars, Phobos. Both examples explain the analysis that has been done and how this analysis has been transformed into an interactive system overview that supports communication of the system's behavior. The examples have not been validated with domain experts as the main objective of the examples is to show the application of COMBOS in various domains. The objective has not been to actually design an electric vehicle or space mission.

7.1 Electric Vehicle

As a first example, an electric vehicle was chosen because it provides interesting behavioral aspects that significantly influence the eventual design. The quantifications used and concepts selected in this examples are based on adaptations of existing data [Figenbaum *et al.*, 2015, Husain, 2011, Tesla Motors, 2015]. Finally, this section is structured following the steps of the method presented in Figure 6.2 and concludes with a discussion in section 7.1.4.

7.1.1 Step 1: Characterizing the Overall System

In order to be able to give an example of a behavioral analysis of an electric vehicle, it is necessary to define a basic system structure initially. As described in section 6.3.1, this involves identifying the systems context and subsequently establishing a provisional system architecture.

Figure 7.1 shows a context diagram for the electric vehicle at an overall system level. The aim of this context diagram is to identify various stakeholders and quantify their relationships. To model all aspects of a system is impossible. In this case the choice is made to focus on systems costs and concepts relevant for charging the battery. Using the information that is provided by modeling the context, the concerns of the system can be identified as well. An overview of the concerns is given in Figure 7.2. When looking at this

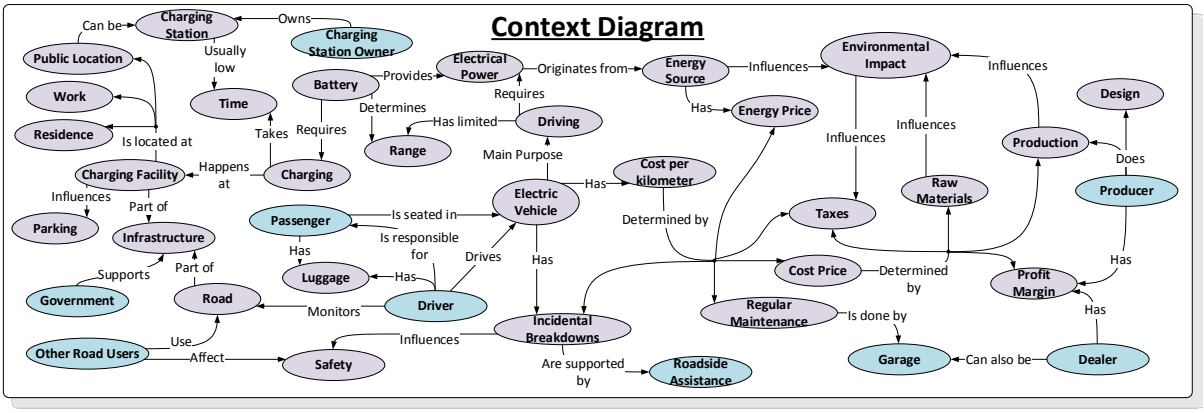


Figure 7.1 – Overall context diagram for an electric vehicle

overview, it can be seen that it resembles concerns for a regular car. However there is an emphasis on for example driving range, because electric car range is currently limited and infrastructure is still lacking. Another example is environmental impact, which is a main driver for electric cars whereas for regular cars this is often an afterthought.

7.1.2 Step 2: Creation of Structural Views

The next step in COMBOS is to develop structural views for each concern. However, as the example is mainly meant to demonstrate the communication of system behavior, only the concern that is used to illustrate the behavioral analysis is developed.

Normally, once a structural view on each of the subsystems and key drivers is obtained, the need for additional analysis can be identified. For an electric vehicle there are various behavioral analyses that can support the conceptual system design. To uncover these needs, the triggers presented in section 6.3.3 can be used. For example, an analysis that can help to gain insight in system cost is to model the effect of various infrastructure and government involvement scenarios. Such a model is described in [Figenbaum *et al.*, 2015] and specifically explores the behavior of the systems context. Another analysis, focusing on

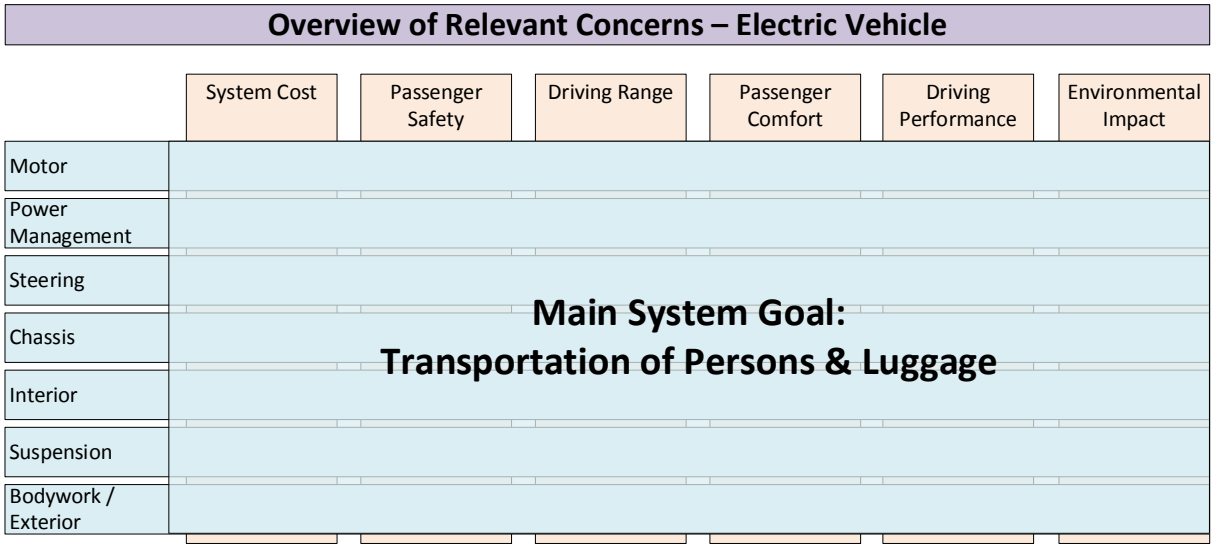


Figure 7.2 – Concerns overview for an electric vehicle, with key drivers (top) and subsystems (left)

external variation could be the driving range. The driving range of a car depends on a number of external factors, for example ambient temperature, traffic conditions and driving style of the owner. Finally, there are system states that require exploration in for example power management. While the main function of an electric vehicle is to transport persons and luggage, modern vehicles include much functionality that support the user during driving or for example provide entertainment.

Similar to the use case presented in section 5.3, it is important to explore the various systems states to determine which parts of the system are available at what time, especially since the available power is finite. To continue the example, it was chosen to elaborate the power management subsystem. The role of the power management subsystem is to store, use and preserve energy, in order to support the overall system goal, transporting a user to a desired location. The overview that describes the structural views is presented in Appendix H.3. It includes a functional view, physical view, quantification view and a context diagram. Note that the operational view and the visual aids were established later, when constructing the behavioral views.

7.1.3 Step 3: Creation of Behavioral Views

The third step in COMBOS is the creation of behavioral views. In order to do so, the detailed steps presented in Figure 6.3 will be followed.

Step 3A – Identifying the Need for Behavioral Analysis

By choosing to elaborate the power management subsystem, it is implied that there is a need for behavioral analysis. The communication of system behavior has the goal to verify the behavior of the power management subsystem across systems states. In order to support these design discussions, the following analysis goals have been established:

- Characterization of the influence of the available battery capacity on system behavior
- Identification of availability of vehicle features during operation of the electric vehicle
- Identification of allowed transitions between the various states of the vehicle
- Characterization of the most influential factors on battery capacity

Step 3B – Behavioral Characteristics

The behavioral analysis is started with the question whether the behavioral characteristics of the system are clear. In the structural analysis, it has been identified that the main power usage is caused by either the use of the engine, use of the air conditioning, or by long periods of inactivity. However, a number of contextual factors were identified that also influence the power management. These are for example weather conditions, such as temperature and wind. Furthermore, driving behavior and driving conditions such as traffic jams can have an influence on the amount of acceleration and deceleration that is used, which in turn affects the power usage. Finally, the distance and nature of the trip that is planned by the user will affect the power management as well.

The effects of the various factors that were just mentioned have clear characterizations on the power usage of the system. Of course "unknown unknowns" (see section 2.4.1) may exist. However, it can be stated that the behavioral characteristics are quite clear within this example.

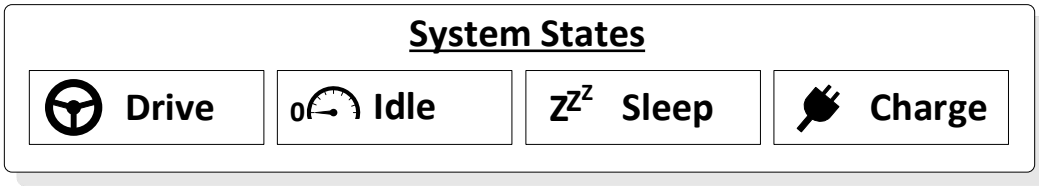


Figure 7.3 – Overview of selected system states for electric car example

Step 3C – Identify Relevant Systems States

Therefore, the next step is to identify relevant systems states. In this example, the user's perspective was chosen to consider the states. For every system, it should be considered which stakeholder's perspective is most relevant when making design decisions.

By investigating the user perspective through considering possible operational modes, various system states were identified. For example, a user can be in the car, or leave it and park it somewhere. When a user is in the car, it can either be driving or standing still and when the car is parked, it could also be charged at the same time. In the end, four main system states are envisioned. These are driving, idle, sleep and charging. An overview of these system states is given in Figure 7.3.

However, within these states distinctions can be made as well. Reasoning from the users' perspective, driving an electric car can be done for maximum performance, maximum battery life or maximum range while driving. These distinctions within states can either be considered as separate states, or later used as variables that control a simulation. In this case, it was chosen to use the second approach.

Step 3D – Detail System Overview for each System State

This stage encompasses detailing the separate views of the A3 Architecture Overview for each of the four system states. The detailing of the physical view can be seen in Figure 7.4.

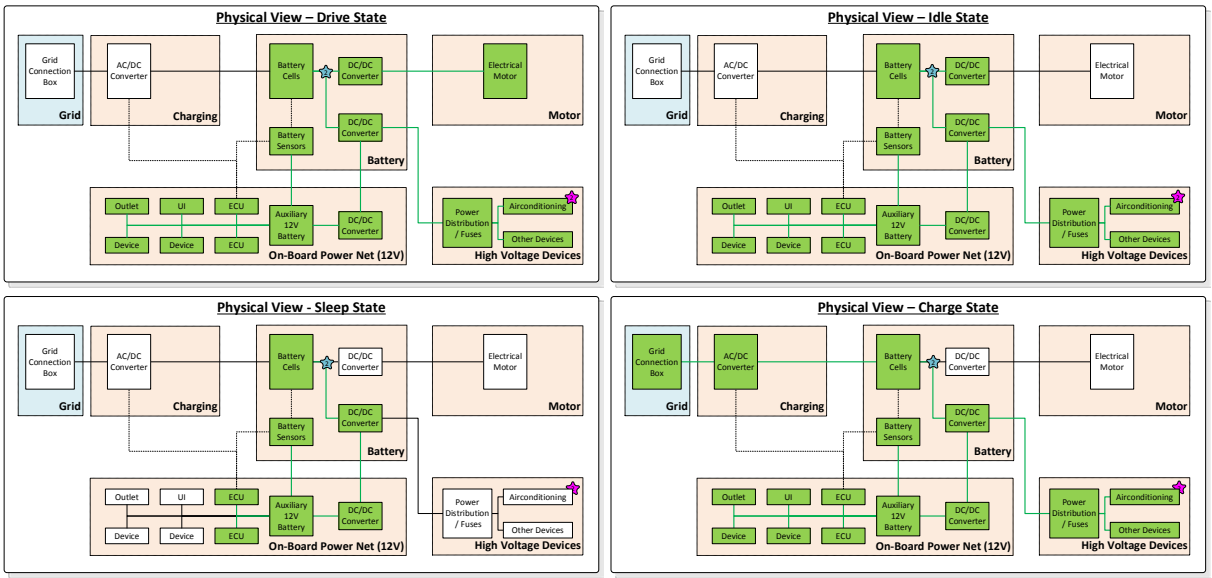


Figure 7.4 – The physical view detailed for the identified system states: (clockwise) Drive, Idle, Charge and Sleep. Larger versions can be found in Appendix E.1

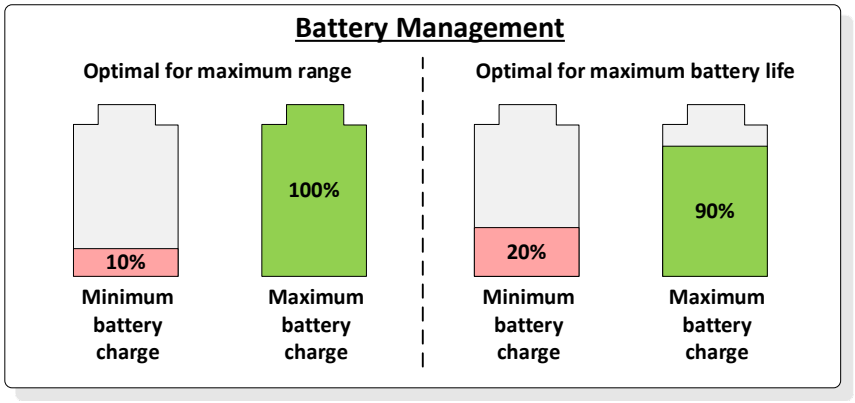


Figure 7.5 – Visual aid explaining battery management

In this case, a green color indicates which parts of the system are powered in that specific state, both in connections and parts.

The same procedure was repeated for the functional view and the context view. The resulting views, as well as larger version of the physical views can be found in Appendix E. The quantification view remains the same for each system state. This was done due to the fact that most of the parameters currently specified in that view do not change in the states that were considered. However, a few of the parameters are interesting to consider over the system's states. These are battery capacity (and range), as well as the power loads of the engine, the air conditioning and the stationary loads. These aspects were incorporated directly into the simulation, as well as portrayed in visual aids. For example an important concept in battery power management is the balance between maximum range and maximum battery life. This issue was explained in a visual aid, which can be seen in Figure 7.5.

At this point, the states are considered in further detail. For example it was already stated that the "Drive" state can have either an economy, performance or range mode. These modes also correspond to Figure 7.5, as economy and performance modes will still strive for maximum battery life, but the range mode will override this behavior.

Step 3E – Construct System Behavior Simulation

A simple approach to construct a system behavior simulation would be to create a state transition simulation, where all transitions are allowed. However, one of the goals of the analysis is to show the influence of the available battery capacity on system behavior. This means that the state transitions should consider this parameter. Also, it is directly clear that transitioning from the charge state to the drive state might run into some issues, such as driving away while the charger is still plugged in. As a first step to structure the simulation, a state transition diagram was created. This diagram can be seen in Figure 7.6.

In this diagram various state transitions and conditions are shown. The diagram shows that the Idle state acts as the central state from which all other states can be accessed. Also, within the drive state, the system can be either in the economy, performance or range mode. While in drive mode, the vehicle can be switched between these sub-modes. This is also possible in charge mode, as the system charges the battery completely in range mode, but not in other modes. This diagram does not specify the transition conditions between

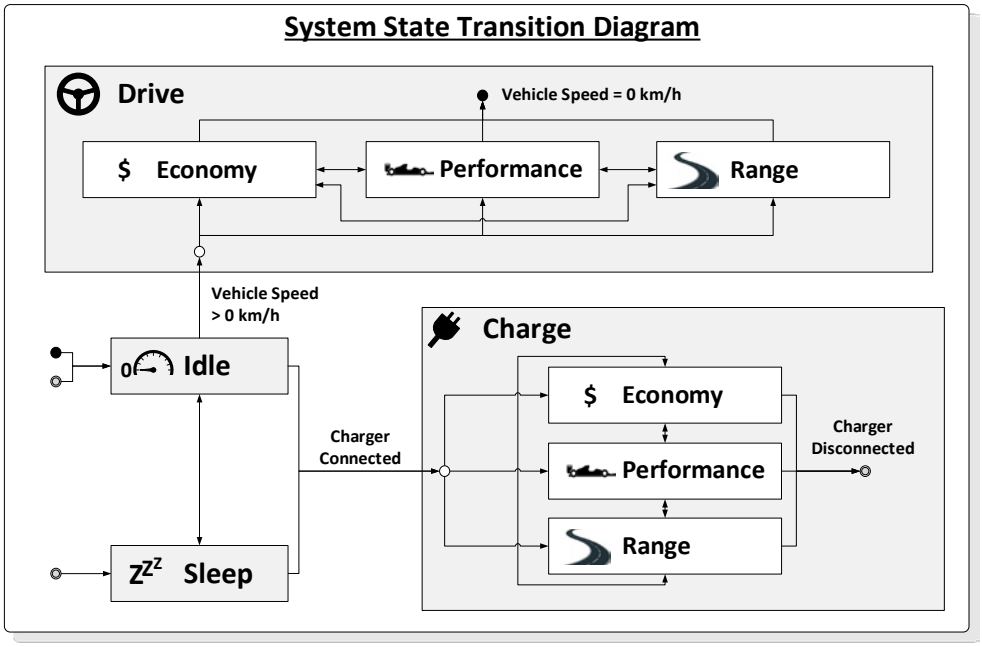


Figure 7.6 – State transition diagram for an electric vehicle

these states, but it can for example be envisaged that with a low battery capacity, the transition from the economy mode to the performance mode is not allowed while driving.

Discussing the state transition simulation forces a formalization of the states and transitions, reducing ambiguity [Hooman *et al.*, 2012]. This formalization helps design discussions, because it for example questions how a state transition condition should be defined exactly and how this can be achieved. For example, the transition condition from drive to idle with a speed of 0 km/h raises the question how accurate the sensor measurements must be to guarantee a true standstill of the vehicle. Or maybe it has merit to consider a certain time threshold in which the car speed must be 0 km/h, for example a few seconds and only then transitioning. Another option would be to transition when the user activates an (electronic) handbrake.

An actual simulation was been developed for this example, but the same approach used in the case study in section 5.3 can be used, as is also described in [Schuts and Hooman, 2015a, b].

Step 3F – Create UI with System View and Visual Aids

The final step in establishing a basis for the communication of system behavior is to integrate the simulation into an interactive system overview. For example, the user needs to have a possibility to change the system state. In this case, it was chosen to provide the user with buttons that directly indicate the state. This can be seen in Figure 7.7. Another option is to visualize the vehicle with a physical representation, and for example allow users to press a virtual accelerator pedal. This could trigger the system to go the driving state.

Available states can be selected by clicking the corresponding radio button. The drive and charge states have additional options, which can be used to control usage scenarios and which will influence the battery capacity in the simulation. The charging options are based on the parameters identified in the quantification view. Both of these options serve

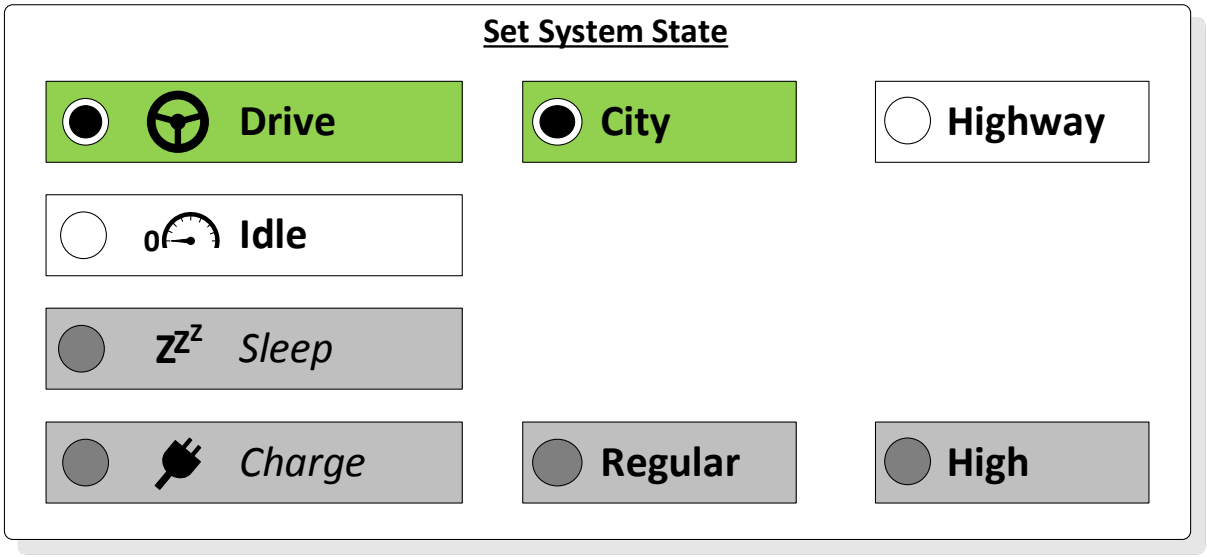


Figure 7.7 – Example of selecting states. System is in Drive state, and can only transition to Idle state. Sleep and Charge state are greyed out (unavailable)

as an indication that different behavior is possible within these states. Of course, for example the drive state could have been made extra complex, by adding driving styles, amount of traffic on the road or specific geographic properties such as driving in the mountains.

While detailing the systems overview and constructing the state transition simulation, a number of important contextual influences were identified. These were amongst others battery capacity and the driving modes. Also, the influence of the temperature on battery capacity and air conditioning usage was considered to be relevant. The state transition conditions already indicated that battery capacity influences availability of these state transitions. Therefore it makes sense to allow the user to observe and adjust the battery capacity to experience its effect on the system behavior.

The same holds for outside temperature and desired inside temperature, as well as the driving modes. Therefore, a number of user controls were introduced to enable the user to interact with the simulation. These controls are illustrated in Figure 7.8.

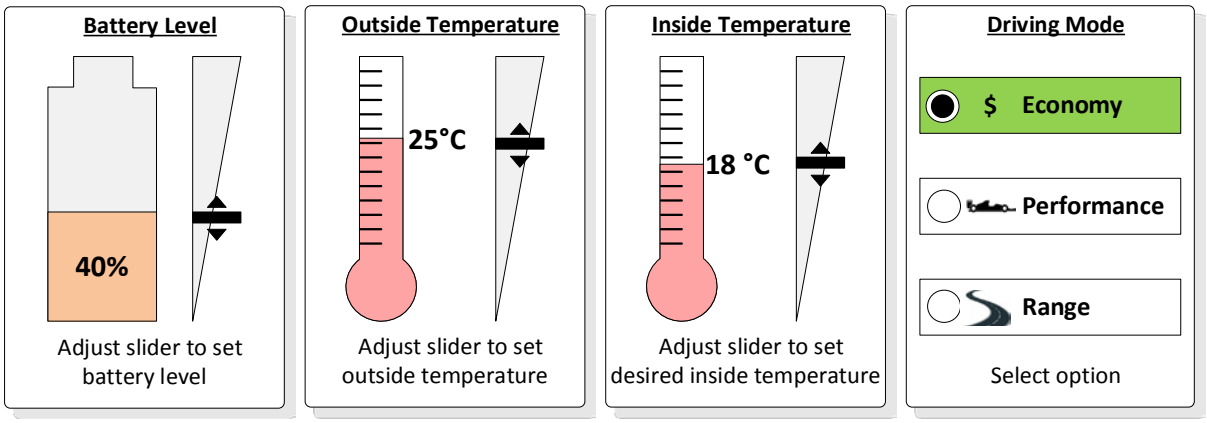


Figure 7.8 – User controls to adjust simulation parameters

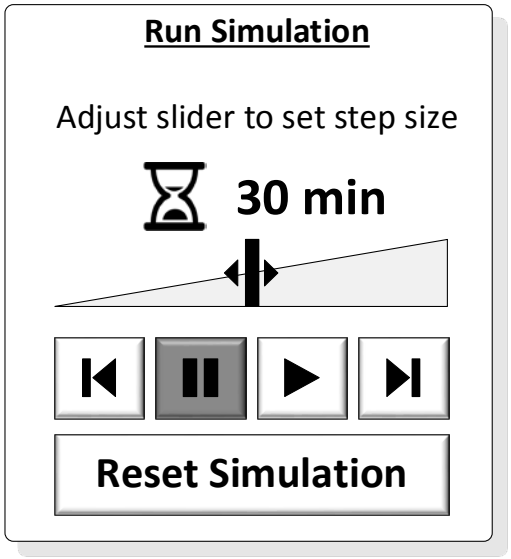


Figure 7.9 – User controls to run simulation. The user can set step size, go a step backward, pause, play, go a step forward or reset the simulation

The final component in the user controls is to let the simulation advance through time in a specific state. This for example influences on the battery capacity or the inside temperature that might change based on the operation of the air conditioning. This progress is made visible to the user by updating variables and states that are changed with a visual indication (blinking). In this case, it was chosen to allow the user to choose a specific duration and run or advance the simulation with that step size. The user controls to do this can be seen in Figure 7.9.

While the simulation is running, battery capacity will change based on the system state as well as outside temperature, desired inside temperature and driving mode. As it is currently envisioned, scenarios are executed manually. However, automated execution of predefined scenarios is also possible. Aside from the user controls in the simulation, visual aids can be used as well to illustrate certain behavioral issues.

For example, it can be beneficial to show the driver view on battery capacity and available range. Therefore, a visual aid that depicts this driver feedback was introduced, see also Figure 7.10. This visual aid allows stakeholders to understand the feedback that the

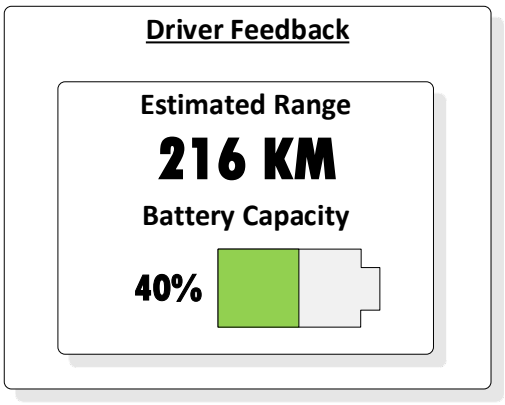


Figure 7.10 – Visual aid showing driver feedback

driver receives. This visual aid is updated alongside the simulation.

The combination of all these views, simulation controls and visual aids results in a final interactive system overview. This interactive system overview is shown in Appendix H.3.

7.1.4 Discussion

As has been stated in the introduction of section 7.1.3, the goal of the system overview is to allow design discussion on four topics. A review of these topics will give some insight in the usefulness of this system overview and whether this goal was achieved.

Influence of the available Battery Capacity on System Behavior

The user interface gives users of the interactive system overview the possibility to set a specific battery capacity and subsequently explore the various system states. For example,

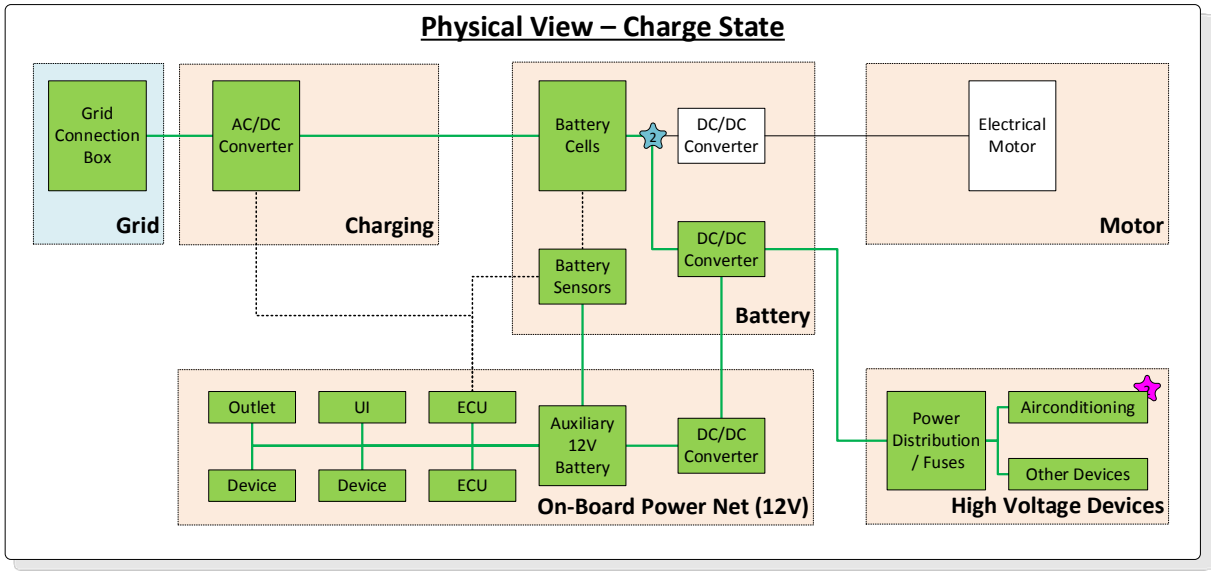


Figure 7.11 – Physical view for the "Charge" state

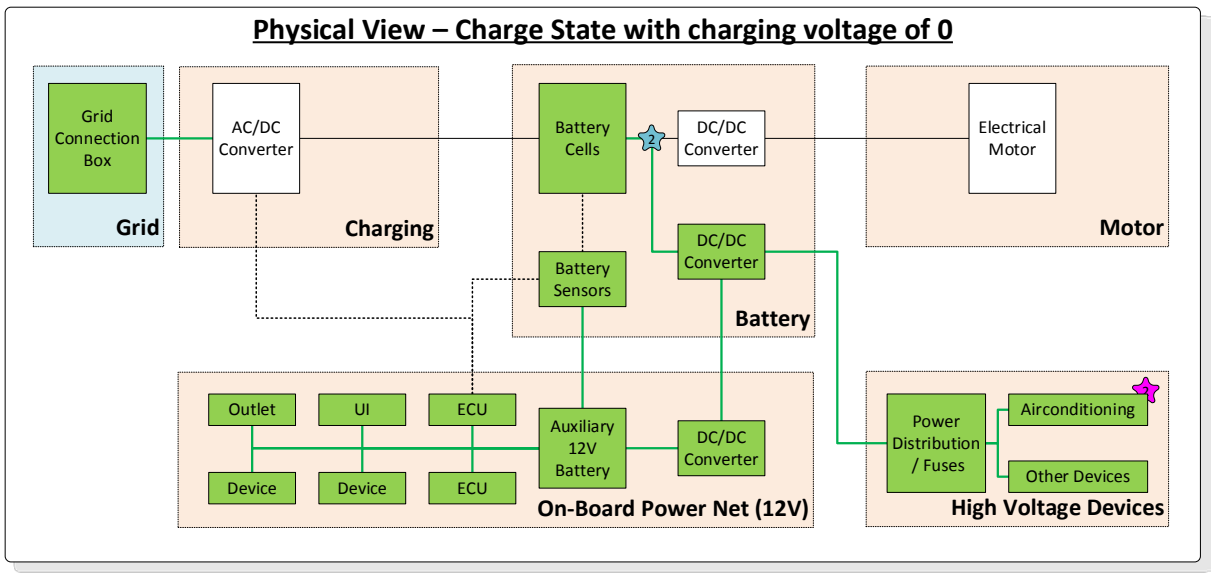


Figure 7.12 – Physical view for the "Charge" state with charging current set to zero

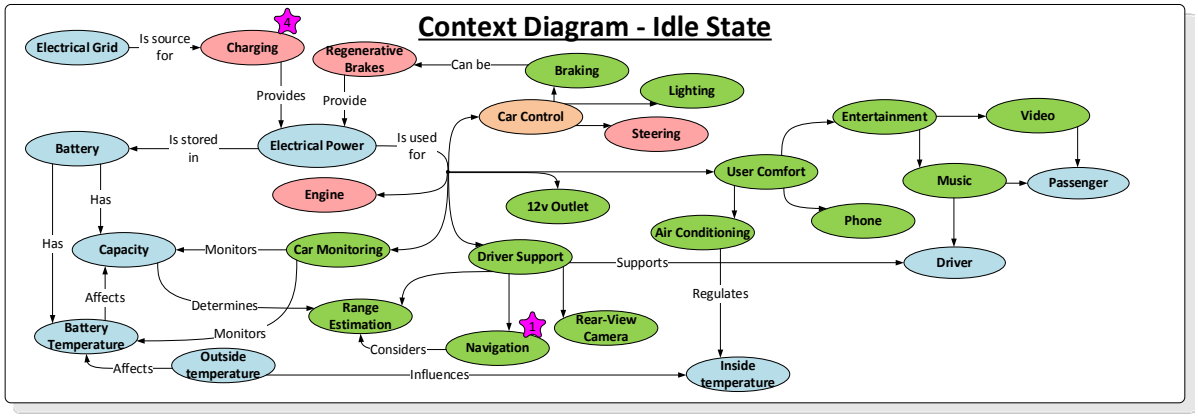


Figure 7.13 – Context View for the "Idle" state of the electric vehicle

when setting a battery capacity of 90% and selecting the charge state, the system can be viewed from various perspectives. The physical view in this state is shown in Figure 7.11. However, from the functional perspective in the same state it is known that there is a function active in this state termed “set charging voltage”. This led to the realization that there might be situations in which the charge current may be lower, or even completely absent in this charge state. This has of course implications for the physical view. For example, if the battery capacity reaches 100%, the charging current needs to be zero to prevent overcharging of the battery. Another example is when the vehicle is either in economy or performance mode, the maximum (advised) battery capacity is 90%. When reaching this threshold, the current is also zero. In these cases, the physical view is different than in Figure 7.11. The correct physical view is shown in Figure 7.12. In this view it can be seen that the AC/DC Converter and the connection towards the battery cells are unpowered.

Availability of Vehicle Features during Operation of the Electric Vehicle

By being able to see the functional view, but especially the context view of the system, it can be assessed which vehicle features are available in which state. For example, the context view in the system idle state indicates that most of the vehicle functionalities are available, as can be seen in Figure 7.13. This view allows discussion whether or not all of those functionalities are desired in that state, or if maybe functionalities can be disabled to for example save power. For example it can be seen that under the subgroup “car control”, lighting and braking is still available, but steering is not powered.

Allowed Transitions between the Various States of the Vehicle

In the current example, the insight in transitions between system states can be gained based on experimenting with the simulation to see whether or not the simulation allows a transition and in what manner. After that, it can be discussed whether this behavior is desirable or not given the state of the system and contextual factors. This is shown in Figure 7.7, where unavailable states are greyed out.

Characterization of the most Influential Contextual Factors on Battery Capacity

The quantification view shows, amongst others, the most influential loads on battery capacity. Furthermore, the simulation has incorporated the most influential contextual factors (driving state, driving mode and temperatures). When developing the simulation, these mathematical characteristics are embedded in the simulation and cannot be analyzed directly by the user. However, the user can observe the system behavior, which for example expresses itself by a fast battery drain in cold weather. From an engineering perspective, it could have merit to make the mathematical relations explicit. If the driver perspective is considered, the driver is only interested in an indication of the available range, though an indication of the current power consumption could be helpful in this regard as well.

Additional Issues

To continue on the concept of iterations, it must be made clear that the current system overview explaining both system behavior and structure is the result of an iterative process, even though this has been described in a linear manner in this section. The iterative process in itself is a result of the interaction with stakeholders, whose views, ideas and questions are represented in the overview. In this example, as it has been devised by the author of this thesis, the interaction has only been present by alternating between viewpoints

To conclude this section, this example shows that a similar application to the case study discussed in section 5.3, but in a different field, can also give insight in the dynamic behavior of a system. The overview created is able to support communication and convey aspects of the system behavior that were determined to be relevant upfront.

7.2 Phobos Sample Return Mission

The second example application presented in this chapter focuses on the space industry. The specific case that is considered is a concept study towards a sample return mission from one of the moons of Mars, Phobos. This case study was selected because it has been a concept study that has been conducted within the concurrent design facility (CDF) at ESA (for a discussion of the CDF at ESA, see section 2.1.3) and is publicly available [ESA, 2014]. This provides an adequate data source to construct the example. A further reference used to construct this example is [Kemble, 2006].

Finally, as opposed to the previous example discussed in section 7.1, this example will not focus on a subsystem but on a key driver. It also starts with a simulation model as a basis to uncover behavioral characteristics, instead of identifying the system states and constructing the operational view based on the systems states.

7.2.1 Step 1: Characterizing the Overall System

Once again, the COMBOS process is started with characterizing the overall system. The overall context for the Phobos sample return mission can be seen in Figure 7.14. In this context diagram it can be seen that a spacecraft needs to travel from earth to Phobos and back again. Furthermore, the spacecraft is supported by a structure that contains a number of elements necessary to complete the mission. The mission itself has a cost, duration and an associated operational risk. This operational risk is increased by the fact that the

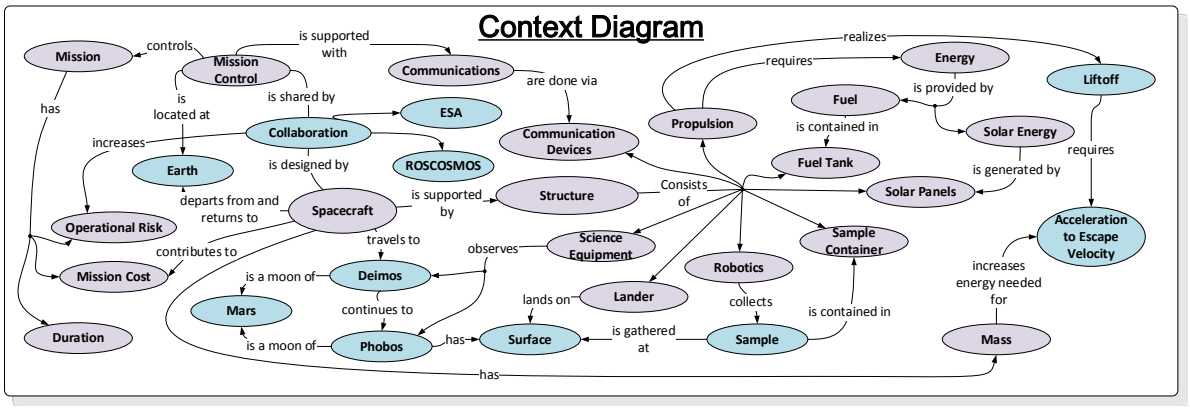


Figure 7.14 – Overall context diagram for the Phobos sample return mission

mission is a collaboration between ESA (European Space Agency) and ROSCOSMOS (Russian Space Agency). Finally, the spacecraft itself has a mass, which influences the energy needed to accelerate to escape velocity. Concepts that were excluded from this context diagram are durability and reliability, as well as the conditions in space and the earth's atmosphere that needs to be entered and exited. Based on this context diagram, as well as the information given in the study report, the relevant concerns for the mission can be identified.

This is visualized in Figure 7.15 and is fairly generic for a spacecraft. Delta-v defines the required changes in speed, which ultimately determines the required amount of propellant. The main goal of the mission is to return a sample from the surface of Phobos to earth. However, the robotics subsystem, which would normally be only used to deploy a payload, now actually is required to perform a complex operation, the collection of a surface sample.

7.2.2 Step 2: Creation of Structural Views

The next step in COMBOS is to define the structural views for each of the subsystems and key drivers. However, this is once more only done for the example that is further explained

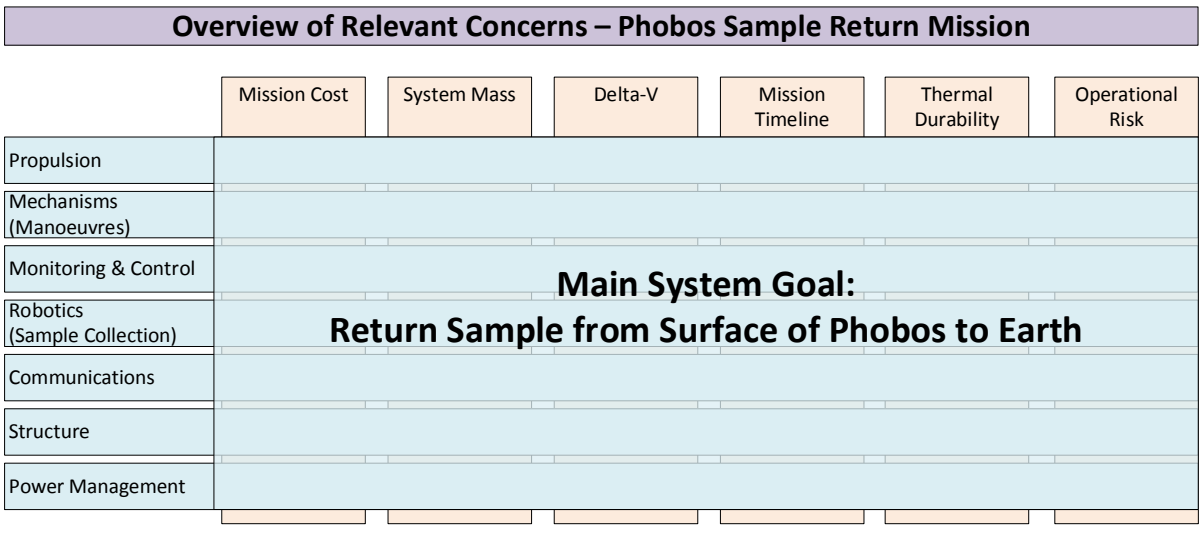


Figure 7.15 – Concerns overview for a Phobos sample return mission

Table 7.1 – System states during operation of the mission

Launch and Direct Escape	Descent and Landing Phase
Transfer Earth-Mars	Surface Operations Phase
Transfer to Deimos	Ascent Phase
Deimos Close Proximity Phase	Departure Phase
Transfer to Phobos	Transfer Mars-Earth
Phobos Close Proximity Phase	Re-entry Phase

in this section. To identify a need for further analysis, especially into the system's behavior, the triggers discussed in section 6.4.1 can be employed.

Considering the mission goal that is proposed, it can be deduced that this mission will have a lot of stages. The stages, or system states that were ultimately identified in the study report [ESA, 2014] can be seen in Table 7.1. Evidently, it is important to characterize both the systems structure and its behavior across these system states for various subsystems and key drivers.

Another issue for behavioral analysis can be found in the characterization of the mission timeline. The mission timeline depends on the spacecraft velocity during transfers. However, it also relies on a correct timing of the transition to a next phase, which is only possible during certain time windows. For example, when the spacecraft has landed on Phobos, communication is only possible about three times per earth day in two hour slots. Furthermore, detailed data will be obtained during operation that can call for a need to refine or change the systems configuration. Therefore, there is uncertainty in the time it takes to properly configure the system for a next stage, which in turn could affect the timeline.

It was chosen to conduct a behavioral analysis for the key driver "mission timeline". The scope of the mission timeline is the time from lift-off to returning on earth. Development and manufacturing time or scientific analysis afterwards are out of scope.

The created structural overview can be found in Appendix H.4. It includes a functional view which is based on the various stages the mission will follow. It also includes a rough physical view and a quantification view that details key parameters as well as a quantification view that gives an initial estimation of duration of all stages. Finally, various visual aids were introduced. These further detail various stages of the mission, or explain some relevant concepts and terminology such as conjunctions, Hohmann transfers and launch windows.

7.2.3 Step 3: Creation of Behavioral Views

The third step in COMBOS is the creation of behavioral views. In order to do so, once again the detailed steps presented in Figure 6.3 will be followed.

Step 3A – Identifying the Need for Behavioral Analysis

The fact that the mission timeline was chosen implies that there is a need for behavioral analysis. The main goals of the communication of the system behavior of the mission timeline are to allow stakeholders to:

- Identify and explain the main issues that determine the mission timeline
- Understand how these issues together influence the mission timeline
- Gain insight in the impact of mission timeline on the overall system

Step 3B – Behavioral Characteristics

The first goal of the analysis is to uncover and identify the main issues that determine the mission timeline. The initial structural analysis indicates some of these issues, such as the available launch windows and possible conjunctions during the mission timeline. However, how they affect the mission timeline and what the specific interactions between these factors are, is unclear. Therefore, it is necessary to first conduct a more in-depth simulation study to uncover the most important characteristics.

Step 3G – Conduct Simulation Study

To analyze and characterize the behavioral characteristics, a simulation model was developed using the Y-chart methodology [Kienhuis *et al.*, 1997]. In the model, the application view is represented by the mission stages and the platform view is represented by (the parts of) the spacecraft.

The process followed to characterize the application view starts with an identification of the different kinds of behavior that the mission stages exhibit. These are grouped into classes, and these classes are characterized with relevant parameters and behavioral aspects. The mission stages, which were already identified in Table 7.1, were grouped into three classes. These are transfers, maneuvers and operations. The same process is followed for the platform view, or the spacecraft components. However, only a single class was established, termed "spacecraft element". The class groupings and the definitions of these classes are described in Appendix F. Based on these classes, a system model for the mission timeline can be defined by defining system elements and quantifying them.

The mapping of application elements to platform elements defines the system model. This mapping is also bound to certain rules that can be incorporated in the model. For example, an application element defines the required thrust for a maneuver, being either high, low or none. A platform element also specifies the type of thrust it is able to deliver, which is either high, low, mixed (high & low) or none. In this way, it can be verified whether or not a platform element is capable of performing the mapped application functionality.

The application elements (or mission stages) have a duration and based on launch windows, dates can also be estimated during which these stages actually take place. This also allows a check to see whether or not a mission stage falls within an Earth-Mars-Sun conjunction and adjust the mission timeline according to this. [ESA, 2014] mentions that a 50-day period will be introduced during a conjunction in which the spacecraft must be in a safe orbit and is unable to perform critical or regular operations.

In the current class implementation, the estimation of the duration of a single application element is done inside the application view. This means that the mapping has no influence on the mission duration when the system is specified with this level of abstraction. This requires for example specifying the distance in the application view, instead of the duration, and specifying the speed of a platform element. The mapping relates a distance in

an application element with a speed of a platform element. After dividing the distance by the speed, this results in the duration.

The current class definition does allow various other explorations. For example, it is possible to calculate the amount of propellant that is necessary to complete the mission. This is relevant, because the mission timeline can be adjusted to reduce the use of propellant. This can be done by specifying the delta-v required, the dry mass and the propellant type with a specific impulse for each instance in the application view. Calculating the required propellant can be done using the Tsiolkovsky rocket equation [Tsiolkovsky, 1903], which rewritten to yield the wet mass results in equation 7.1.

$$m_w = m_d * e^{\left(\frac{\Delta V}{(I_{sp} * g_0)}\right)} \qquad (7.1)$$

Here, m_w is wet mass in kg, m_d is dry mass in kg, ΔV is the delta-v to be achieved in m/s, I_{sp} is the specific impulse in seconds and g_0 is the acceleration due to gravity at the Earth's surface (9.81 m/s^2).

The model as described has been implemented in Excel [Microsoft, 2010a], with the ability to specify application and platform elements according to the class definitions. This model outputs a date-based projected mission timeline that accounts for conjunctions. The timeline also incorporates a delay caused by these conjunctions and indicates the margin in days before the departure transfer date. Furthermore, it estimates the total amount of propellant required for the mission, resulting in a total wet mass required to execute the estimated mission timeline. The implementation of this model allows exploration of different mission timelines.

Step 3H – Extend Simulation with System View

After constructing the initial model to uncover the behavioral characteristics, the simulation can be extended to fit the communication about the system's behavior. Currently, the model allows a user to identify the impact of conjunctions on the mission timeline by varying the launch dates of the outbound and return transfers. It shows that conjunctions do not impact the mission timeline significantly in most cases as these fall in a phase where the spacecraft is in Phobos orbit and it is possible to introduce the 50 day safety period without much complexity. However, when the outbound launch takes place in 2028, the conjunction will take place during the landing sequence on Phobos if the regular durations are used for all mission stages. As this is a critical phase, the mission timeline will have to be severely adjusted for a launch in 2028. Therefore, users should be able to adjust the launch dates as well as the mission timeline and it is logical to include these in the system view.

The resulting system view can be seen in Appendix H.5. Several changes were made compared to the initial structural overview in Appendix H.4. Both the functional view and physical view were updated to represent the application view and platform view of the Y-chart model respectively. The application view also includes the mapping to the platform elements. Furthermore, the mission timeline estimation provides more information, such as the dates for each stage. Another addition is that a wet mass calculation is visible and

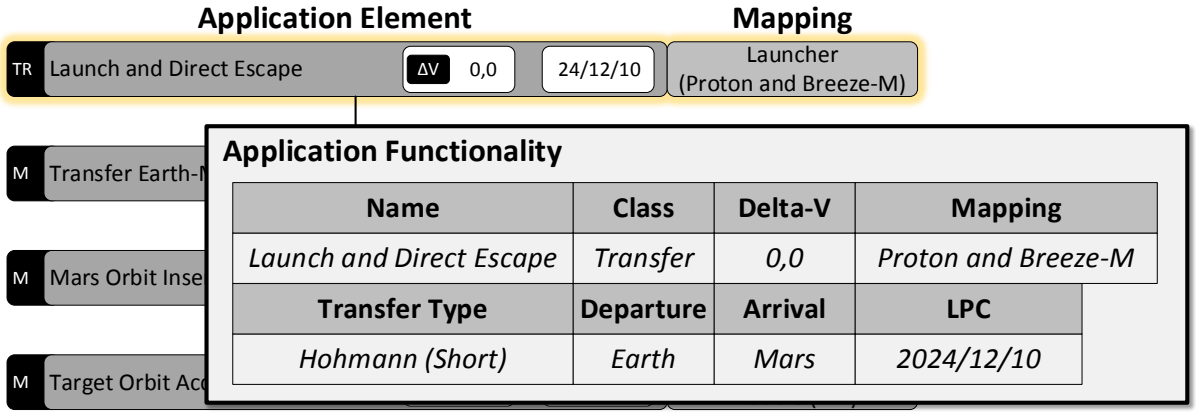


Figure 7.16 – Right-clicking an element in the application view brings up a sub-menu to edit the selected application element

linked to the physical view. Furthermore, the visual aids were restructured to focus on the main mission timeline influences.

The envisioned user interactions with the interactive system overview are the following. Users are able to adjust parameters, change the mapping and the class of application or platform elements. This can be done by left-clicking the visible parameters, or by right clicking an element, which results in opening an extra menu window in which further details can be edited, as is shown in Figure 7.16. When a user changes a parameter, all connected parameters are updated based on the underlying simulation model.

Until now it was not discussed how to realize this interactive system overview in a software implementation. The user interface with the system overview can be implemented using Eclipse. For the simulation model, there are three options. The first option is to use the Excel model as a basis and connect Eclipse to this Excel model. The second option is to implement the simulation model in Java and the final option is to develop a POOSL model to execute the simulation model. Due to the fact that the model is currently fairly basic, it does not need a large number of simulation runs to characterize the system's behavior. Considering this argument, a POOSL model is less useful. However, if there already is an identified need to further detail the simulation model, a POOSL model could be become applicable again. The choice between a full implementation in Eclipse and a connection to the existing implementation in Excel has to be made based on factors outside the scope of this example, such as the existing structure in the organization and capabilities of involved employees. If there is a policy or tendency to use Excel models as central artefacts in the design process, it makes sense to contain the model within Excel and only implement the user interface in Eclipse. This reduces a bit of flexibility, since the interface between the two software environments must be established and maintained. If the model is converted to Java code in Eclipse as well, this allows better portability and consistency.

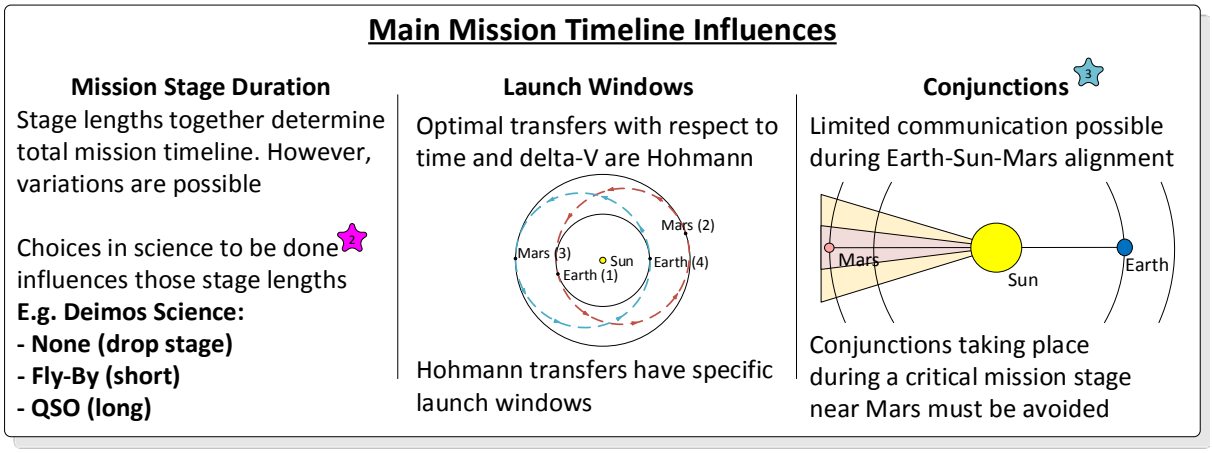


Figure 7.17 – Visual aid explaining the main mission timeline influences

7.2.4 Discussion

The discussion of this example focuses on the main goals of the communication of the system behavior, as they were defined in the beginning of section 7.2.3, as well as several additional issues.

Identify and Explain the Main Issues that determine the Mission Timeline

The analysis led to identification of three main influences on the mission timeline. These three influences are mission stage duration, available launch windows and earth-mars-sun conjunctions. These issues were presented and discussed in a visual aid in the final system overview. This visual aid is also shown in Figure 7.17.

Understand how these issues together influence the Mission Timeline

The mapping of mission stages as application functionality to spacecraft elements as platform functionality allows an exploration in which the launch windows and mission stage durations can be changed as desired within the available bounds. The model will also detect conjunctions during the mission and increase the stage length of the stage in which the conjunction starts. Therefore, users have a good overview of how the identified issues influence the mission timeline.

Gain insight in the impact of Mission Timeline on the Overall System

Next to being able to meet the mission's science objectives, one of the main impacts of the mission timeline and the mission stage planning is the delta-v budget, and subsequently the wet mass of the resulting spacecraft element. By mapping delta-v budgets per mission stage to spacecraft elements with an estimated mass and specific impulse, the impact of adjusting the mission timeline on the overall system is shown.

Additional Issues

Currently, the number of different mission timelines is limited based on the fact that they are constrained by launch windows. Therefore it is not as important to present the outcomes of multiple system definitions at the same time for comparison. If the number of variables and possible mission timelines increase, the overview is easily lost and an

aggregation of results becomes valuable. However, this is also the point where the nature of the activity changes from understanding the system's behavior towards a full design space exploration with the intent to find an optimal design. In that case, an approach such as advocated by [Grogan *et al.*, 2015] has more merit.

To take the next step in detailing the variation existing within the mission timeline and to further support the design space exploration, a number of steps can be taken. An example concerns the definition of the duration of a mission stage exclusively in the application view by an estimation of the number of days. Instead, it is possible to specify the distance in the application view and the speed in the platform view, as was discussed earlier. Another possibility is to calculate the required delta-V for transfers and several maneuvers directly based on the launch windows. A model can be developed that calculates delta-V budgets based on the launch windows.

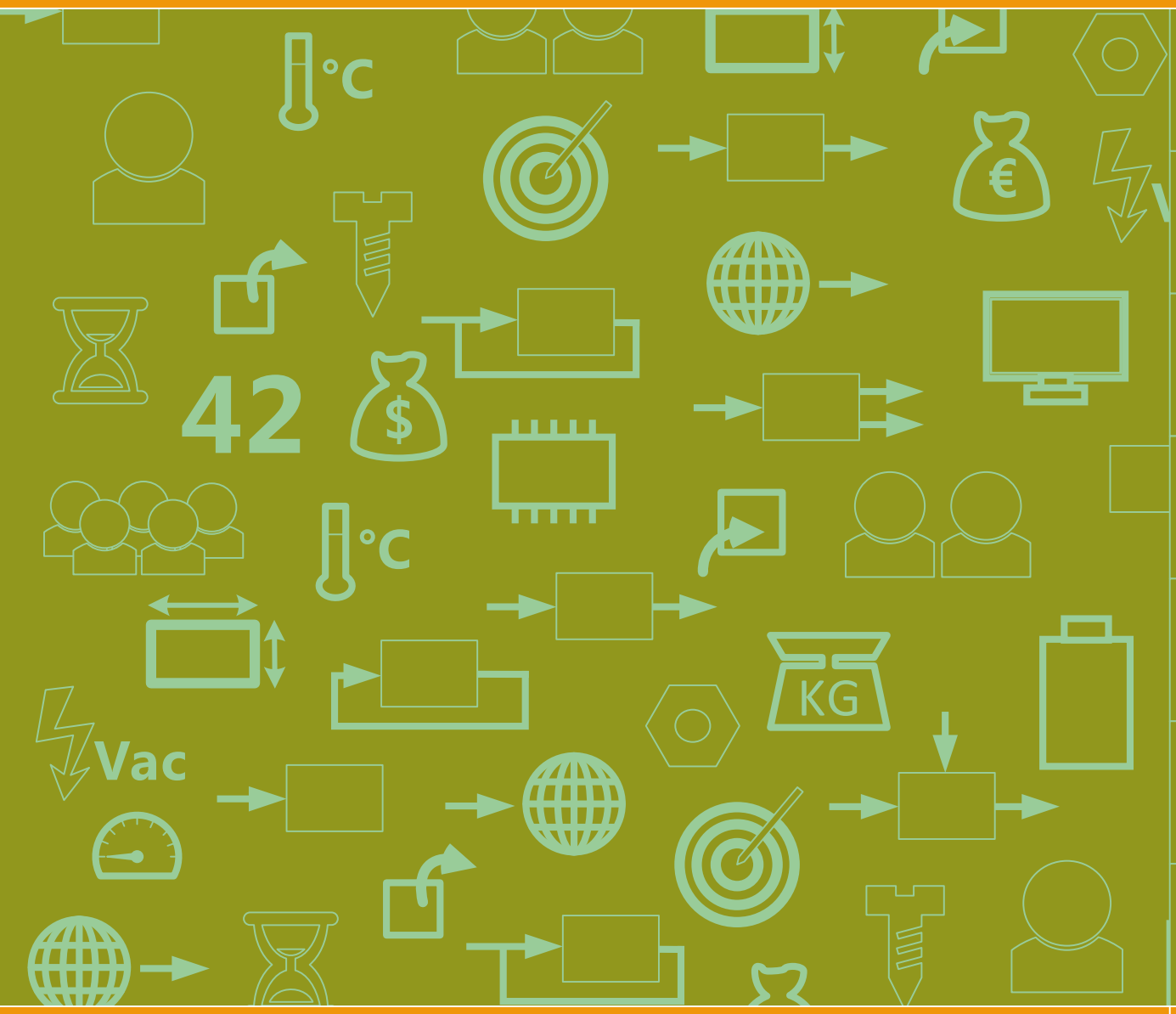
A final discussion point is the Concurrent Design Facility (CDF) which was used to perform the feasibility study towards a Phobos Sample Return Mission as presented in [ESA, 2014]. This facility aims towards multidisciplinary collaboration in short iterations, as was discussed in section 2.1.3. The CDF currently does not have a tool in place that specifically aims to support the communication. This is all done in traditional ways, by making reports and presentations. The CDF does include the OCDT, which is the basis for their Model Based Systems Engineering approach. In section 7.2.3, three options for the simulation model were discussed (Excel, POOSL or Eclipse). A fourth option would be in this case to couple the system overview to the OCDT database, from which it can draw its information. Coupling a system overview to communicate system behavior to a Model Based Systems Engineering model is further discussed in section 8.3.4.

7.3 Conclusions

In this chapter, two examples were given on how to construct a system overview that supports communication of system behavior following COMBOS, as presented in Chapter 6.

The construction of the examples was done within two weeks for each of the cases, even though the researcher was unfamiliar with the respective fields. It must be noted that for both of the examples, a full realization was not obtained. The expectation is that this takes an extra two weeks of work. Based on these examples and the case studies in Chapter 5, the required effort can be estimated. Using the COMBOS method, an initial version of an interactive system overview could be provided within three to four weeks by a single person. The actual time depends on familiarity with Eclipse & the simulation modeling tool, the domain and adaptation of the method, as it requires the capability of information extraction and abstract reasoning.

The first example showed that it is possible to characterize the power management of an electric car across different states and in different perspectives at the same time. The second example showed the other route in Figure 6.3 which aimed to identify the behavioral characteristics and explain their influence on the issue at hand as well as on the overall system. While the examples were not fully implemented and validated in the field, they clearly show that the application of the method in this way is feasible in multiple domains.



Evaluation of the COMBOS Method

8

Evaluation of the COMBOS Method

This chapter comprises the evaluation of the COMBOS method presented in this thesis. The evaluation is started in section 8.1 by reviewing the method based on challenges in communication of simulations defined in Table 4.2. The method is evaluated based on experiences in practice, including a survey with stakeholders who were involved in the case studies in section 8.2. Additionally, section 8.2 reviews the supportive technology that is currently available and obstacles to implement the method. Finally, section 8.3 discusses both the scope of the method and possible extensions.

8.1 Evaluation based on Key Challenges

This section reviews the COMBOS methods in light of the four main challenges in communicating simulations and system behavior that were presented and discussed in Chapter 4. Moreover, these challenges were also used to define the requirements for the intended support in Table 5.1.

8.1.1 Accommodate Multiple Disciplines

In conceptual system design, the communication process should aim to accommodate multidisciplinary views, to allow a broad audience to understand and discuss the issues at hand. Where possible, jargon should be avoided and multiple views should be supported.

COMBOS uses A3 Architecture Overviews [Borches, 2010] as a basis for the creation of multidisciplinary views. In this manner, an accessible set of views is provided. Due to the fact that no specific formalism, e.g. SysML [Object Management Group Inc., 2012], is used to formalize all concepts, accessibility is increased, and the use of jargon is minimized.

One of the main goals of COMBOS is to provide an overview of system behavior with multiple different views at the same time. This allows users to explore specific states of the system from multiple perspectives and allows identification of inconsistencies between views or new possibilities that are identified by cross-linking these views. Furthermore, transitions between states or changes in behavior can be observed from multiple perspectives at the same time, as has been discussed in section 5.3.

8.1.2 Support Divergent Design Space Exploration

Most design space exploration tools [Grogan *et al.*, 2015] focus on supporting users in finding an optimal solution to a problem. However, in conceptual system design it is important to explore the problem domain properly, before focusing on the solution domain.

COMBOS focuses on exploring and explaining behavior, and less so comparing solutions. In the space mission example presented in section 7.2, two different mission timelines cannot be viewed at the same time. The overview focuses on showing the impact of various

system characteristics on the overall system behavior, and does not aim for an optimization of the mission timeline directly. If optimization were the goal, a database to store results of previous simulation runs and a means to retrieve and show these results alongside one another is required, allowing for manual optimization. While this is valuable, this should be considered the next step in the process, after applying this method. Of course, automated optimization strategies can be applied as well. The fact that the method uses the Y-chart methodology allows a transition to this kind of design space exploration [Basten *et al.*, 2013].

Divergent design space exploration also implies that new types of behaviors or system designs can be explored. Furthermore, it raises the question whether the method supports the discovery of emergent behavior. It was experienced that two parts are required to support this. First the modeling and simulation approach and tools used must offer flexibility and secondly, the method must steer stakeholders towards discussions that stimulate out-of-the-box thinking. The first part is enabled by the fact that the method emphasizes an abstract, system-level approach of modeling. Furthermore, the POOSL language and its tools offer flexibility in modeling that is required for this type of application. The second part is enabled by focusing the communication and overviews on the key system characteristics and their impact on the resulting system behavior. This avoids the solution oriented approach that many design space exploration tools exhibit.

8.1.3 Dealing with Uncertainty

During conceptual system design, uncertainty surfaces in various forms and shapes. This can be either uncertainty in a parameter, uncertainty in the systems functionality or even uncertainty in the systems context. Systems architecting and systems engineering are activities that must make decisions under uncertainty. Various approaches have been reviewed that allow dealing with uncertainty in various ways, through mathematical approaches [Rajabalinejad and Bonnema, 2014], formalized review techniques [NASA, 2008] or by identifying and quantifying various parameters to obtain a simple mathematical based decision [Thompson and Paredis, 2010].

While all of these approaches can be applied to cope with uncertainty, most decisions are still made implicitly by stakeholders, based on available information and gained experiences. COMBOS aims to strengthen this informal decision making process by increasing the insight of stakeholders in a subject that is hard to quantify, the system's behavior. This is done through continuous communication and stakeholder involvement, aiming to represent all stakeholders' concerns. Discussing and representing these concerns increases stakeholder confidence and ultimately will reduce uncertainty.

COMBOS thus reduces uncertainty based on increased stakeholder communication and representation of their perspectives to capture the "known knowns". The method also helps to elicit the "unknown knowns", by forcing stakeholders to consider their knowledge from different perspectives and realizing that some of their knowledge might be relevant for the design. Connecting knowledge from different perspectives also helps to explain behavior of the system, which results in a better understanding of observed phenomena. COMBOS also helps to identify the "known unknowns", because stakeholders are forced to explicitly repre-

sent open ended issues in the system overview. The final type of uncertainty in this reasoning is "unknown unknowns", which can partly be uncovered by the overall system-level view used to model system aspects. This was for example experienced with the "outlier" behavior in the hand-eye coordination case study discussed in section 5.2. The method does not support dealing with "unknown unknowns" that can surface during a system's operation, as it does not focus on the robustness of systems specifically. However, the chosen modeling approach using POOSL and domain specific languages [Schuts and Hooman, 2015a] allows for formal verification approaches [Keshishzadeh, 2016] that help to uncover errors in a design, increasing a system's robustness.

8.1.4 Lack of Formality

Conceptual system design and multidisciplinary communication are informal by nature, while simulations require a significant measure of formality to operate.

COMBOS shows that a balance between the two can be achieved, by approaching the simulation process with a high-level system model and specifying only what is necessary. This allows the simulation to connect well to the interactive system overview that is used in the communication of system behavior, as both rely on the same abstraction approach. The method employs a fairly loose coupling between the simulation and system overview. In the electric car example in section 7.1, the inner workings of the simulation model can be derived from the interactive system overview. Only the state transitions are not explicitly visible in the example, but they can be inferred through exploration of the interactive system overview. In the space mission example in section 7.2, the simulation and its underlying model are more hidden. For example, mathematical relationships are not made explicit. This is due to the fact that the choice was made to abstract from the underlying simulation model and focus on other aspects in the interactive system overview.

COMBOS has a few drawbacks as well, for example scaling of the method is not straightforward. In the power management subsystem of the iXR system in section 5.3, 12 states are discussed, and the actual case study considered 24. In practice, the number of cases can be further increased for the underlying POOSL model. However, dealing with this increased number of states is not something that is supported well with this approach, as visualizations have to be created for each state. However, this is not considered as a huge drawback, given that conceptual system design should consider a limited number of states to ensure meaningful design discussions.

8.2 COMBOS in Practice

This section discusses the experiences with the implementation of COMBOS and the resulting interactive system overview in practice. This is done via a survey with involved stakeholders, a discussion of required technology and a discussion on how the method can be embedded into an organization.

8.2.1 Stakeholder Experiences

To enable a formalized way of collecting qualitative feedback of individuals that have been involved in the creation or usage of interactive system overviews, a survey has been con-

ducted. Appendix G details the full survey approach and presents detailed results. This section summarizes these results and discusses their outcomes. The survey was completed by two system architects and two domain specialists that were involved in the case study presented in section 5.3. It touched upon various subjects, such as the use of simulation models, the use of interactive system overviews, communication of system behavior and conceptual system design.

Regarding simulation models, all respondents acknowledged that these are important in conceptual system design. A system architect stated: *"simulation models are becoming a must to manage complexity"*. Furthermore, the respondents indicated that they could not properly estimate whether connecting a simulation model with an interactive system overview was too much work for the added value. A system architect noted: *"it does seem like a lot of work to a non-software designer"*. However, in general they did not view the connection to interactive system overviews as very complex. A domain specialist stated: *"what seems complex is the "thinking" needed to carry this activity out, but that should be done regardless of the tool used"*. With regard to system behavior, they acknowledge the support offered by seeing changes in the system behavior across different views. All respondents expect that the interactive system overview will improve communication, within their own discipline and outside of it as well.

The overall conclusion from the survey is that the respondents acknowledge the support that the approach as well as the resulting interactive system overview offers. However, they are unsure about the amount of work and technical skill required for implementation of the approach. Therefore, they are currently unable to judge the value of the approach in comparison to existing practices.

8.2.2 Supportive Technologies

To be able to apply the method in practice, used applications and devices must be available. COMBOS uses Eclipse [Eclipse, 2015] as a modeling environment and the POOSL language, which is available as a plugin to Eclipse [Embedded Systems Innovation by TNO, 2014]. Eclipse is an open source framework and the POOSL plugin is open to use as well, with proper acknowledgements. Therefore, the software development as described is readily available. [Heer and Agrawala, 2007, Heer and Shneiderman, 2012] support the notion that the technology and interaction used for visualization are not the most complex issue. The complexity lies in selecting the views that are presented in the visualization.

However, to share and use interactive system overviews, devices with sufficient screen size are required as well. Earlier research [Melching, 2012] and especially [Brussel and Bonnema, 2015] led to the realization that maintaining the A3 size in interactive A3 Architecture Overviews is important to preserve the overview. The current interactive system overviews are developed for a minimum resolution of 1920x1080. This can be either computer screens for desk usage or in small meetings, or large screens or beamers that support this resolution during larger meetings. Unfortunately, in this manner, it is not possible to use the interactive system overview similar to a regular A3 overview, which is to have it lying on a desk to be picked up at a moment's notice. This might prove a large threshold to use the interactive system overview.

Therefore, it was explored whether the usage of A3 tablets address this threshold by using the interactive system overview on a portable tablet computer, the Dell XPS 18 [Dell Corporation, 2015]. Experiences with this tablet show that it allows similar use situations to a paper based-A3, due to the fact that it is easier to group around it than a PC, by laying it flat on a table. Furthermore, in large group settings the tablet can be passed around to share control of the interactive system overview that is shown on screen. It this sense, a tablet retains some qualities of the paper based A3. However, the fact that it is not paper-based increases the threshold of simply picking up the overview for a quick discussion. Also, the portability of this tablet computer is significantly worse than a simple A3 paper. This results in a reduced availability of the interactive system overview. Future technology developments such as lightweight large screen tablets, possibly using e-ink, or foldable or rollable displays could help to lower this threshold [Brussel, 2014].

8.2.3 Embedding the Method into a Company

Next to obstacles imposed by technology, the company developing a complex system will also encounter several barriers to apply the method and interactive system overview due to its organization.

One of the main obstacles from this perspective will be maintenance and ownership of an interactive system overview. Much like the A3 Architecture Overviews, the interactive systems overview should have a clear owner, preferably the system architect. If the company is large enough and employs multiple system architects, system architects should be responsible for the interactive systems overviews that deal with their part of the system. The fact that a system architect is the tool owner should ensure consistency between different system overviews. Multiple projects with different timelines will exist at the same moment. As a result, it is difficult to ensure consistency in a formal way. A strategy to increase consistency could be to couple this method to Model Based Systems Engineering, which is discussed in section 8.3.4. The general advice would be to have a set of views per project describing the system architecture in terms of subsystems and key drivers. Where necessary to explain system behavior, these views should be interactive. System architects should analyze and compare views from different concerns to ensure consistency.

The threshold for adoption of the method lies mainly in whether or not a company recognizes the value of having system level overviews of particular or all system aspects. The results of the survey with involved stakeholders (see Appendix G) show that they expect that the method can be applied on case by case basis, which allows an easy introduction.

8.3 Extending COMBOS

This section discusses the method in a broader sense, comparing it with other methods, tools and approaches. It also discusses possible connections to these methods and extensions to the method itself.

8.3.1 Architectural Scope

COMBOS currently aims to create an interactive system overview that supports communication on the behavior of one specific system concern. As was discussed in Chapter 3, the system architecture in the conceptual system design phase can be described with a set of views for the overall system concern, and a set of views for each subsystem and key driver. The method currently aims to create a set of views that covers one single concern, not connected to other concerns. This is useful when considering this single concern in detail, especially when focusing on the system's behavior with respect to that concern.

However, it is possible to connect the interactive system overview of a single concern to the overall system view. In this case the effect of the behavior of one concern on all other system concerns can be viewed. This can be achieved by explicitly visualizing a FunKey diagram [Bonnema, 2008] in the interactive system overview and modeling the connections to the parameters in this diagram. However, it was concluded that it is more beneficial to explicitly visualize the aspects that are really impacted, and omit the others, as space in the system overview is fairly constrained. The interactive system overview of the space mission example (section 7.2.3) clearly visualizes the impact of the mission timeline on system mass and delta-v. In contrast, the impact on thermal durability is omitted, as this key driver is not influenced as strongly.

A further consideration can be to connect an interactive system overview to interactive overviews of other system aspects. This allows linking and moving between views, and ensures consistency. Without additional support, this is difficult to achieve. If a supporting framework in the form of Model Based Systems Engineering is available, it could be considered. Connecting the method to a MBSE framework is discussed in section 8.3.4.

8.3.2 Extending the Domain

Currently, the method is used in the healthcare domain and examples in the automobile and space domains are presented. The simulation models behind these case studies and examples are fairly similar, especially the comparable healthcare device power management case study and electric vehicle example. This raises the question whether the applicability of the method is determined by the domain or by the model used. Clearly, differences exist in the way the interactive system overview is visualized when comparing the electric vehicle example in section 7.1 and the space mission example in section 7.2.

The electric vehicle example concerns the power management subsystem. The power management subsystem represents a functional concern. This type of concern can be described well with a state transition simulation. In this situation, the underlying goal of the interactive system overview is to explain the system behavior by showing it across a number of states. On the other hand, the space mission example considers a key driver, the mission timeline. This key driver represents a performance concern. In this case, a model characterizing this performance needs to be developed first. Using the modeling approach of the Y-chart methodology allows users to adjust both parameters and the system's structure and to observe the resulting effects on the overall system behavior. In general it can be said that concerns about functionality benefit more from the approach used in

section 7.1 while performance based concerns often require the approach used in section 7.2. These are the two paths presented in Figure 6.3, where section 7.1 follows the direct path from 3A to 3F and section 7.2 follows the process via 3G and 3H.

In essence, it can be stated that because the method is applied in the conceptual system design stage, the applicability across domains is fairly large. If the same concepts can be identified in other domains, it is possible to apply the method. The approach used in for example the hand-eye coordination case modelled the throughput of an image processing system. Similar concepts in other domains would be a bottle filling line in a brewery in the food industry, patient flow in a hospital or traffic using a highway system.

8.3.3 Model Types

The previous section already briefly touched upon what kind of (simulation) models are supported by the method. Currently, the method has been implemented using discrete event simulation, which as explained in section 5.2.2 was deemed most suitable to support behavioral analysis in the conceptual system design stage.

Applying other types of simulation strategies has not been a subject of research in this work. For example agent based or physics based modeling approaches are also applicable to systems engineering. Agent based modeling strategies are mostly used in the exploration of System of Systems. Physics based modeling approaches are used more in detailed design. Physics based simulations can be described similarly to the Y-chart modeling approach, while an agent based modeling simulation will have commonalities with a state transition simulation based on discrete event modeling. The agent based modeling simulation has the complication that multiple systems must be considered at the same time, while the interactive system overview only has been designed to communicate about a single system.

Virtual reality applications, used in for example serious gaming applications cannot be directly connected to COMBOS. However, 3D models can be used to represent the physical view, as is for example realized by [Hooman *et al.*, 2012]. [Thalen, 2014] describes lightweight approaches to utilize virtual reality in the conceptual system design stages. This allows the inclusion of these kinds of representations in the interactive system overview. Value modeling is an interesting approach to consider as well. Value modeling focuses its efforts on formalizing a system architecture into a single value metric, something that seems to violate the informal multidisciplinary approach of the method presented in this thesis. However, it was experienced in [Balk, 2015] that value models cannot rely on only presenting this single value, but require a means to give stakeholders insight into the value model. In this sense, it is interesting to consider this method for that particular application, although a straightforward implementation is not readily available.

A final consideration is when multiple model types are required to describe behavior of a system aspect. In the case studies and examples, a single model type was enough to analyze the behavior for a specific concern. However, if multiple model types are required, various approaches can be used. [Derler *et al.*, 2012] discuss an approach to model heterogeneous systems using Ptolemy II. [Basten *et al.*, 2013] present the Octopus Toolset, which is based on the Y-chart approach and also supports multiple models. They

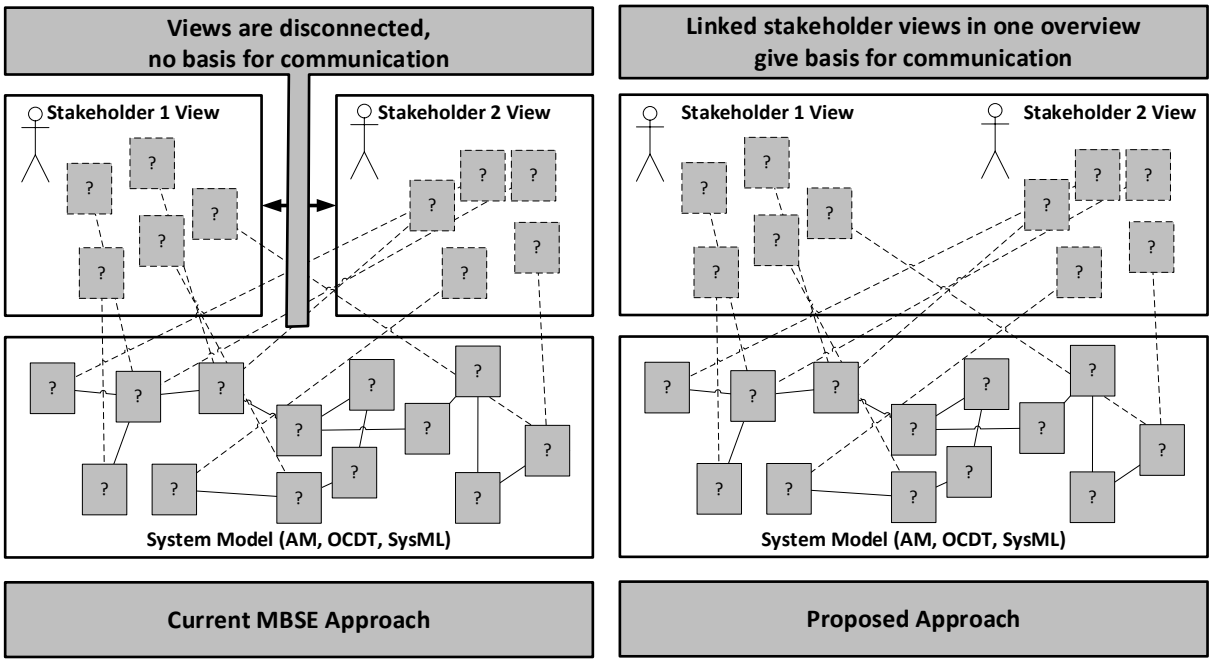


Figure 8.1 – How the method in this thesis can support Model Based Systems Engineering

also refer to Metropolis [Balarin *et al.*, 2003] as another possibility to use integrated models from multiple domains.

8.3.4 Model Based Systems Engineering

In section 8.3.1 it was stated that if there is a desire to connect all information to ensure consistency, an underlying framework is required. MBSE applications such as the Architecture Model [Woestenenk, 2014], SysML [Object Management Group Inc., 2012] and OCDT [Koning *et al.*, 2014] formalize all concepts in a system and link them in a database. This helps to ensure consistency across the specification of the system design. MBSE imposes a rigid structure that requires a lot of effort and commitment to adhere to. However, when followed correctly, it brings great benefits.

The relation with the method presented in this thesis is that this method can help to support the MBSE process. Figure 8.1 shows how this is envisioned. On the left side, the classic MBSE approach is shown. A central database contains the system model, in which concepts are linked in a certain structure. Each stakeholder is able to construct his own view based on the concepts in the system model and the data associated with those concepts. If a stakeholder updates the data in the concepts that are part of his view, these will update as well in the system model and subsequently in the views of other stakeholders. This ensures consistency. However, as the views are selected and generated separately, there is no basis for communication other than the fact that the model is based on the same data. By allowing stakeholders to create their own views, presumably in their own tools, it is hard to link the extracted views of both stakeholders. COMBOS forces the views to be part of the same overview and describe the same system state. In this way, a better basis for communication can be achieved.

Using interactive system overviews as a front-end to present the information contained in the MBSE system model on a particular system aspect has already been explored. For example, hyperlinked A3 Architecture Overviews based on a MBSE database explaining the safety concern of a high speed train line were established in [Schuitemaker *et al.*, 2015]. However, this implementation does not yet consider the representation of system behavior in an interactive manner.

8.4 Conclusions

In this chapter, the COMBOS method presented in Chapter 6 was reviewed and discussed from various perspectives. It was shown that the method addresses the challenges that exist when utilizing simulations in early, conceptual system design. Furthermore, initial application of the method showed that stakeholders in general recognize the value of the method in communication of system behavior during conceptual system design. However, they are currently unable to compare this value to existing practices, mainly due to uncertainty in the actual effort required to apply the approach.

In addition to this, the method was discussed in a broader perspective, pinpointing the current architectural scope of the method, as well as its applicability in various domains. Finally, possible connections and extensions to different model types and Model Based Systems Engineering was discussed.

9

Discussion

This chapter provides a reflection on the conducted research that has resulted in this thesis. This is done by discussing and reviewing the contributions of this thesis, including a reflection on recent developments that share commonalities with this research. Finally, the research approach is reviewed.

9.1 Contributions

This section defines and reflects on the contributions that are realized through the research presented in this thesis.

9.1.1 Support for Use of Simulations in Conceptual System Design

The literature review presented in Chapter 2 to Chapter 4 identifies that simulations are underutilized in problem exploration in conceptual system design, as they are mainly used to search for optimal system configurations. Furthermore, the review shows that communication support is an area that is severely lacking in current descriptions and frameworks for simulation studies. Four challenges that need to be addressed when using simulations in system design (see Table 4.2) were identified. A framework was presented to address these challenges (see Figure 4.2).

In summary, because the literature review emphasized a lack of communication support, a framework was developed to support application of simulation techniques in conceptual system design. This allows both future research and industrial developments to apply simulation techniques in conceptual system design more easily.

9.1.2 Reference Model to Communicate Modeling and Simulation Activities

In Figure 4.3 a reference model for communication of modeling and simulation activities was presented. This model links key concepts related to modeling and simulation activities to the conceptual model that is presented in the ISO/IEC/IEEE 42010 standard [ISO/IEC/IEEE, 2011]. The reference model provides an overview of the relevant concepts and aids both research activities as well as development activities. This is achieved by using the reference model as a checklist when performing simulations studies and constructing the communication support. The concepts in the reference model serve as building blocks for the communication support, as it specifies the views that can be used. It also supports the process, as it explicitly mentions that the system of interest has a context that impacts its dynamicity. Finally, the concepts explicitly mention that the visualization of the communication should focus on key characteristics of the system and use accessible views that allow group decisions on the system's design.

9.1.3 Method to Communicate Behavior of Systems

The main contribution of this thesis is the COMBOS method, presented in Chapter 6. COMBOS is a process that identifies and characterizes system behavior in interactive system overviews that can be used for multidisciplinary communication.

COMBOS advocates continuous communication during the development process using (interactive) system overviews and through continuous iterations in the modeling process. This modeling process is started by describing the system on an abstract level. The continuous communication is important because one of the main pitfalls in communicating simulations is to only communicate at the start and end of the analysis process. Communication at the end of the process allows other users to explore a simulation once it is finished by varying parameters, changing inputs or by changing the systems context. While much insight can be gained in this way, users will gain confidence in the simulation and the simulation will represent their perspectives better when they are closely involved during the construction of the simulation. The other advantages of continuous communication are the increased capability to find errors in the design, leading to an increased quality of the design. Applying the method should replace the more implicit design discussions on system behavior that are currently in place. However, some additional, but limited effort is expected for creation of the interactive system overview.

Collaboration using simulations is not something that is completely new. For example, conferences dedicate tracks to collaborative modeling and simulation [D'Ambrogio *et al.*, 2014]. However, the simulation methods, tools and applications that are presented by D'Ambrogio *et al.* and in other places in literature often focus on engineering, and less on architecting. Therefore, they consider collaboration to be a data sharing activity. This thesis and the method presented in this thesis considers collaboration, especially in the architecting stage of systems design to be a knowledge sharing activity. The initial experiences with application of the method show that this goal can be achieved as multidisciplinary design discussions on the conceptual system design are supported.

Concurrent engineering applications in so termed "war rooms" [ESA, 2015, Mark, 2002] also mainly regard simulations from a data sharing perspective. However, by colocating multidisciplinary teams, knowledge sharing is enabled by socialization [Nonaka and Takeuchi, 1995]. By providing an underlying Model Based Systems Engineering framework [Koning *et al.*, 2014], the CDF facility at ESA ESTEC also supports the externalization of tacit information. However, this is mainly information that is focused on the solution domain. In comparison to these "war room" approaches, the method presented in this thesis supports externalization of tacit information from both the problem and solution domain using the approach of the A3 Architecture Overview method [Borches, 2010].

A further difference is that COMBOS adheres to a full A3 based interactive system overview to discuss the system behavior versus simply extending the user interface of a simulation with views that might be relevant, as has been done in [Mooij *et al.*, 2013]. While this approach has merit due to the practicality and relative simplicity, it does not have a structured approach for view representation as provided by the COMBOS method. This structure can be especially advantageous when the method is applied to visualize

multiple system aspects of a conceptual system design. In this situation, the entry barrier to access information is lowered due to familiarity with the interactive system overview.

9.2 Recent Developments

During the execution of the research presented in this thesis, which started in November 2011, various new developments have come to light. While some of those earlier developments are already integrated in the research in this thesis, this section discusses these developments and compares them to the research conducted in this thesis.

An example of this is a recently published handbook on modeling and simulation-based systems engineering [Gianni *et al.*, 2014]. Both the fact that this book has been published as well as its contents are relevant to this research. The editors themselves state that

"The capability that modeling and simulation (M&S) supplies for managing systems complexity and investigating systems behaviors has made it a central activity in the development of new and existing systems. However, a handbook that provides established M&S practices has not been available".

They furthermore state that M&S practices are still predominantly mono-disciplinary and that the handbook offers an initial means of cross-domain capitalization..

The book includes chapters on "war rooms", performance engineering, multi agent modeling and also a chapter describing an extension to Object Process Methodology (OPM) [Dori, 2002], called Vivid OPM [Dori *et al.*, 2014]. OPM is one of several modeling approaches that enable abstract reasoning on system behavior in the conceptual system design phase, similar to recent work presented by [Canedo and Richter, 2014]. VIVID OPM is an interesting initiative that aligns with the approach presented in this thesis as it aims to extend the representation of a purely OPM based view with other views. This resulted in an animated view using symbolic descriptions of the concepts described in the OPM model. For example the concept of a bank is described with the picture of a bank building. This reportedly gives stakeholders an increased understanding of interactions in the system.

[Canedo and Richter, 2014] on the other hand use functional modeling connected to a multi-domain physics-based simulation to be able to characterize the system. The modeling approach is similar in nature to the modeling approach presented in this thesis, with the exception that Canedo uses physics-based simulations while this thesis uses discrete event based simulations. Another difference is that Canedo focuses on architectural exploration, while this research focused on knowledge transfer and providing system overview.

Several recent research efforts regarding system overviews have been conducted in the research group of the author. These include research on interactive A3 Architecture Overviews [Brussel and Bonnema, 2015], research on communicating MBSE using A3 Architecture Overviews [Schuitemaker *et al.*, 2015] and providing a system interpretation tool to improve customer insight [Kempen, 2015]. These all indicate the feasibility of using interactive and connected systems overviews as a basis for communication of complex system aspects. In addition to this, [Muller, 2014] showed that a paper based A3 can be used to represent operational views, by using a "comic-book style" of use case description. [Singh and Muller, 2013] show it is possible to connect multiple A3 Architecture Overviews

by hyperlinking them digitally. Furthermore, the application of Knowledge-Briefs (K-Briefs) are described in [Klein *et al.*, 2014, Martinet, 2014]. K-Briefs are also termed A3 reports and share similarities with A3 Architecture Overviews [Borches, 2010]. [Klein *et al.*, 2014] conclude that the K-brief can serve as a user interface for contextual information documented in a context driven access system. [Martinet, 2014] discusses the application of K-Briefs at Airbus Defense & Space. They focus on three key aspects in their approach; make it visible, keep it simple and trust your people. These values are represented as well in the approach advocated by this thesis, by creating simple overviews that allow people to make well informed decisions, while avoiding strict formalizations.

A final area of research is the use of interactivity to communicate and discuss issues. These stem from classic "data models" in a web based environment [Grogan *et al.*, 2015, Ricci *et al.*, 2014], to virtual reality applications [Cloutier *et al.*, 2013, Thalen, 2014] and finally to serious games [Garde, 2013, Grogan and Weck, 2015]. While these expand on already existing practices of multi-criteria decision making and stakeholder participation, they show that interactive models are helpful to unlock this information.

In conclusion, there have been various research activities in the recent years that have similarities with the philosophy applied in this research. Although these approaches have different modeling approaches or application domains, it is worthwhile to further investigate their common similarities and further strengthen this research field that considers communication support using interactive, multi-view modeling applications.

9.3 Research Validation

The research presented in this thesis is a result of a long process in which research objectives and goals have changed multiple times. The initial goal described in the Allegio project plan [Embedded Systems Innovation by TNO, 2011] was to develop tools and techniques for high-level system definition in the context of Model Based Systems Engineering. Research in this context identified a perceived gap between basic estimations and mono-disciplinary models and described requirements for high level models supporting design space exploration in the MBSE context that could address this gap [Haveman and Bonnema, 2013]. This narrowed down into research focusing on the question of why simulations are little used in the early stages of design and an approach was suggested to address this [Haveman *et al.*, 2014]. During this research, it was discovered that the main issue is not using simulations in early design, but their communication [Haveman and Bonnema, 2015a, b]. This finally has resulted in the work presented in this thesis.

The research approach as presented in section 1.5 structures the activities that took place in the context of this research. In that light it is also possible to review these activities, and discuss the validity of the used approach, the data sources and the outcomes over the four phases that were identified in section 1.5 using the Design Research Methodology [Blessing and Chakrabarti, 2009].

The first phase, described in Chapter 1 and Chapter 2 was the research clarification phase. This phase consisted of a literature review and resulted in a set of research questions. These research questions helped to frame the further research.

The second phase, described in Chapter 3 and Chapter 4 was a descriptive study that consisted of an in-depth literature review, supported by observations and interviews in industry. These interviews were free-flow interviews with various stakeholders within the organization of the industrial partner, Royal Philips, as well as several interviews outside of this organization (see Appendix A). As there has been no formal approach described for these interviews, the results of the interviews are only used as examples to support argumentation throughout these chapters. The question is whether the validity of the results is still preserved by embedding the interview results in this way. The first reason for the lack of a formal representation is that the interviews were mainly used as an explorative tool to identify issues. Therefore, no single set of questions has been used throughout these interviews. It thus made less sense to describe these informal, often ad-hoc interviews formally. The risk of this approach is that the researcher bias plays an even larger role than normal, as the validity relies on the researcher performing the interviews and selecting the data objectively. This was partly mitigated by a periodic review and discussion of results with project group members including stakeholders from the industrial partner.

The third phase, described in Chapter 5 and Chapter 6, was a prescriptive study that comprised the execution of two case studies as well the development of a method based on these case studies. In terms of the design research methodology [Blessing and Chakrabarti, 2009, p142], the outcomes of this phase are a design method, presented in Figure 6.2, and several design guidelines, both summarized in Appendix D as well as a design tool, the interactive system overviews. The case studies (see also section 1.5) involved collaboration with practitioners to uncover system issues. The case studies represent real-life examples, increasing the validity of the developed support. Also the fact that the continued involvement of stakeholders and their indications to continue using the support (see survey results in section 8.2.1), contribute to the validity as stated by [Muller, 2009, 2013].

The final phase in the research approach, described in Chapter 7 and Chapter 8, is the second descriptive study. In this phase, two examples have been developed to validate the method. Based on these examples, it is possible to conclude that the COMBOS method is applicable to problem at hand. However, it has not been possible to include a validation on the effectiveness of he method in this research, mainly due to time constraints. The example applications and subsequent review of the method both thus only aim to support validation on the effectiveness of this method. A survey was designed to obtain formalized feedback. It was not aimed towards quantitative feedback because the sample size is too small as only closely involved stakeholders could be included. Rather, it was used for qualitative support of the validity. The survey form was chosen in an effort to minimize some of the researcher bias in this stage.

In conclusion, the validity of the initial descriptive study, which affects the validity of the resulting reference model and identified challenges, is deemed to be sufficient. As a note, both of these results have been presented as peer-reviewed conference papers. The method itself is based on valid assumptions as it was based on experiences in case studies that already closely resembled the method itself. Finally, it can be stated that the method presented in this thesis has been positively validated on effectiveness. However, validation on the efficiency and thus the value of the method requires further research.

Conclusions and Future Research

10

10

Conclusions and Future Research

This final chapter concludes the research presented in this thesis. It does so by reviewing the research questions posed in section 2.6 and giving recommendations for future work.

10.1 Conclusions

To structure the conclusions of this research, they will be discussed in the light of the research questions posed in section 2.6.

RQ1: What is the role of behavioral analysis in the conceptual system design process and which views are required to represent behavior?

Behavioral analysis is a key aspect of conceptual system design. It is of paramount importance to understand how system elements interact with the environment as well as internally. Behavioral analysis supports problem understanding and analysis. Without behavioral analysis, it would not be possible to develop systems that behave in predictable ways and are able to cope with a changing system context. Furthermore during the case study discussed in section 5.2 it was discovered that the inclusion of a context diagram in the system overview supported a better focus on the problem domain and improved the insight on the impact on the rest of the system. This has led to the realization that constructing a context diagram for both subsystem and key driver concerns is preferable.

This thesis identified that behavior of a system should be represented across various essential views. The operational view, which describes a system's behavior, should thus be shown across the functional, physical, quantification and as final addition also the context view (see Figure 10.1). The context view is also added to guideline 3 in Appendix D.

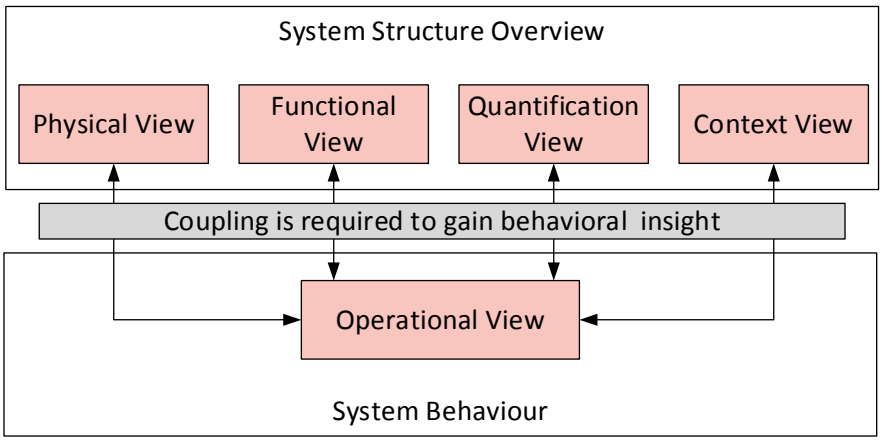


Figure 10.1 – Coupling between views that describe system structure and system behavior, including the addition of the context view (this is an update of Figure 6.1)

RQ2: What are the main challenges that oppose effective communication of simulations in the conceptual system design stage?

The research in this thesis has identified four main challenges that oppose an easy adaptation and usage of simulations in conceptual system design in section 4.2.1. These are (i) accommodating for multidisciplinary views, (ii) supporting divergent design space exploration, (ii) dealing with uncertainty and (iv) a lack of formality. All four challenges have been discussed in depth and possible mitigation strategies have been identified. Finally, the resulting method has been reviewed based on these challenges in section 8.1 and it can be concluded that the method allows users to address these challenges.

RQ3: How are important concepts in communication of models and simulations represented in existing architecture frameworks?

This research has identified these concepts by surveying three key aspects. These are the overall conceptual system design process, the modeling and simulation process and finally the essential views for communication of models and simulations. This resulted in a number of explicit concepts that were only partly represented in architecture frameworks. To gain an overview of these concepts, they were related to the conceptual model of systems architecting in [ISO/IEC/IEEE, 2011] in an reference model (see Figure 5.3). This extended conceptual model provides an overview of important concepts and the relation between them and can function as a mental checklist when communicating models and simulations.

RQ4: What does a method that supports communication of models and simulations look like?

Chapter 6 has presented the COMBOS method which is a process to embed behavioral analysis and its communication in conceptual system design. COMBOS is mainly applicable to single-entity product systems. It considers a conceptual system architecture with views that describes the systems context, functions, key drivers and subsystems as a starting point. It extends this system architecture with a behavioral analysis for key drivers and subsystems, if these are deemed to deliver added value. It supports two approaches. The first one is to use behavioral analysis to show and discuss the system behavior across various states. This is generally related to explaining the behavior of a functional concern. The second approach communicates a behavioral study that aims to understand and uncover behavioral characteristics. These are generally related to performance concerns.

The method provides guidelines and a stepwise process description that can be used to execute COMBOS (these can be found in Appendix D). Furthermore, a supportive tool was introduced in the form of an interactive system overview. This overview, based on the A3 Architecture Overview method [Borches, 2010], enables stakeholders to observe behavior across a number of relevant views, while maintaining system overview.

RQ5: What is the applicability and value of the method resulting from RQ4?

The method as presented was evaluated in a number of ways. First of all, the applicability of the method was tested by applying it to two example case studies. These showed

that the application of the method is possible to various domains and types of problems, although the electric car example in section 7.1 was similar in nature to the power distribution case study in section 5.3. Furthermore the examples showed that it is possible to build an interactive system overview that provides insight in system behavior with limited effort. Second, the fact that the method addresses the challenges was already discussed for RQ3. Finally, a survey was held under stakeholders that were involved in the initial experiences with the method. This survey showed that the method is applicable to the communication of system behavior. Respondents indicated that the method supports communication of system behavior, both within and outside of their discipline. However, they had difficulties to assess the value of the method compared to current practices.

Finally, the contributions of this research were reviewed in section 9.1 and it can be concluded that this thesis has contributed to the body of knowledge with respect to communication of simulations in conceptual systems design, by investigating which challenges exist and how they can be addressed using a proactive approach. Furthermore, in systems design it is often said that the devil is in the details [Muller, 2011b]. This method can help to drive out the devil by allowing stakeholders to communicate and reason about system behavior. This way, the method acts as a piece in the puzzle of improving the overall conceptual system design process. This in turn may lead to systems that can be developed faster, cheaper and above all have the behavior that stakeholders expect and desire.

10.2 Future Research

The research presented in this thesis, offers many possibilities for future work. In this section, recommendations are given for future endeavors.

First of all, additional validation activities should be employed that especially explore the value of the method. These can range from research into the time and effort required across a project team, to a perhaps more automated generation of the interactive system overview. This can be done by improving the tooling to connect simulations more easily with system overviews. Also, further research towards used devices as well as user interactions definitely has merit. However, the most important validation activity would be to apply the method in case studies that require multidisciplinary collaboration, with both engineering and non-engineering stakeholders, including the customer and end user. Guidelines presented in [Houy *et al.*, 2012] could be used for the validation. The example applications and case studies in this thesis have been limited in scope in this respect.

Another interesting line of research is based on the work presented by [Grogan *et al.*, 2015]. In this work, a web-based environment is offered that offers a more flexible way of presenting views than the A3 format does. The interactive system overviews aim to preserve the concept of restricting the presented information to about an A3 sized equivalent. However, the approach of [Grogan *et al.*, 2015] allows hiding views, adding views and using the screen real estate more as a canvas than as a framework. The merit of such an approach is definitely worthwhile to be researched.

Connecting the interactive system overview with existing information databases, such as an AM-model [Woestenenk, 2014], an OCDT-model [Koning *et al.*, 2014] or a SysML model might prove interesting. For example, the interactive system overview could be used as a front-end for communicating information that is contained with an MBSE model. [Schuitemaker *et al.*, 2015] have done some interesting research in this regard. That work, as well as the research presented in this thesis might form a basis for future user interface development for MBSE tool vendors.

Finally, it is worthwhile to dedicate further research to determine strategies to embed the method in organizations. Some relevant adoption barriers were already discussed in section 8.2.3. An interesting research question to start this line of research is whether companies are willing to adopt the conceptual systems architecting views presented in section 3.3.4 as a central frame of reference in their organization. Once this has been established, system overviews, including interactive ones, for communication of complex issues throughout the complete design process can be utilized. A final research direction is to research the value of having a complete set of reference overviews for a system, its subsystems and its key drivers.

Bibliography

Adamsson, N. (2005). *Mechatronics engineering: New requirements on cross-functional integration*. PhD Thesis, Royal Institute of Technology.

Aguwa, C. C., Monplaisir, L. & Turgut, O. (2012). Voice of the customer: Customer satisfaction ratio based analysis. *Expert Systems with Applications,* vol. 39 (11)**,** pp. 10112-10119.

Ahn, J., Weck, O.L. de & Steele, M. (2014). Credibility Assessment of Models and Simulations Based on NASA's Models and Simulation Standard Using the Delphi Method. *Systems Engineering,* vol. 17 (2)**,** pp. 237-248.

Alexander, C. (1964). *Notes on the Synthesis of Form.* Harvard University Press; Cambridge, MA, USA.

Altshuller, G. S. (1997). *40 Principles - TRIZ Keys to Technical Innovation, .* Technical Innovation Center, Inc.; Worcester, MA, USA.

Alvarez Cabrera, A. A. (2011). *Architecture-Centric Design: Modeling and Applications to Control Architecture Generation*. PhD Dissertation.

Arias, E., Eden, H., Fischer, G., Gorman, A. & Scharff, E. (2000). Transcending the Individual Human Mind - Creating Shared Understanding through Collaborative Design. *ACM Transactions on Computer-Human Interaction,* vol. 7 (1)**,** pp. 84-113.

Bahill, A. T. & Gissing, B. (1998). Re-evaluating systems engineering concepts using systems thinking. *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews* vol. 28 (4)**,** pp. 516-527.

Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Paserone, C. & Sangiovanni-Vincentelli, A. (2003). Metropolis: an integrated electronic system design environment. *IEEE Computer,* vol. 36 (4)**,** pp. 45-52.

Baldwin, W. C., Sauser, B. & Cloutier, R. (2015). Simulation Approaches for System of Systems: Events-based versus Agent Based Modeling. *In* 2015 Conference on Systems Engineering Research (CSER), Hoboken, NY, USA.

Balk, R. (2015). *Implementing a Value Model as a Framework for Innovation in Healthcare: Product Development of Test Devices in the Medical Field*. Master Thesis, University of Twente.

Basten, T., Benthum, E., Geilen, M., Hendriks, M., Houben, F., Igna, G., Reckers, F., Smet, S., Somers, L., Teeselink, E., Trčka, N., Vaandrager, F., Verriet, J., Voorhoeve, M. & Yang, Y. (2010). Model-Driven Design-Space Exploration for Embedded Systems: The Octopus Toolset. *In* 4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Greece.

Basten, T., Hendriks, M., Trčka, N., Somers, L., Geilen, M., Yang, Y., Igna, G., de Smet, S., Voorhoeve, M., van der Aalst, W., Corporaal, H. & Vaandrager, F. (2013). Model-driven design-space exploration for software-intensive embedded systems. *In:* Basten, T., Hamberg, R., Reckers, F. & Verriet, J. (eds.) *Model-Based Design of Adaptive Embedded Systems.* Springer, pp. 189-244.

Beek, T. van & Tomiyama, T. (2011). Workflow Modelling of Intended System Use. *In:* Van de Laar, P. & Punter, T. (eds.) *Views on Evolvability of Embedded Systems.* Springer Science & Business Media B.V, pp. 153-170. Eindhoven, The Netherlands.

Berg, F.G.B. van den, Remke, A. & Haverkort, B. R. (2015). iDSL: Automated Performance Prediction and Analysis of Medical Imaging Systems. *In* EPEW 2015, Madrid, Spain.

Birta, L. G. & Arbez, G. (2013). *Modelling and Simulation: Exploring Dynamic System Behaviour.* 2nd ed. Springer; London, UK.

Bjorkman, E. A., Sarkani, S. & Mazzuchi, T. A. (2013). Using model-based systems engineering as a framework for improving test and evaluation activities. *Systems Engineering,* vol. 16 (3)**,** pp. 346-362.

Blanchard, B. S. & Fabrycky, W. J. (2010). *Systems Engineering and Analysis.* 4th ed. Pearson Education International; Upper Saddle River, NJ, USA.

Blessing, L. T. M. & Chakrabarti, A. (2009). *DRM, a Design Research Methodology.* Springer;
Boardman, J. & Sauser, B. (2008). *Systems Thinking: Coping with 21st Century Problems.* CRC Press; Boca Raton, FL, USA.
Boehm, B. (1986). A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes,* vol. 11 (4)**,** pp. 14-24.
Bonnema, G. M. (2008). *Funkey architecting: an integrated approach to system architecting using functions, key drivers and system budgets*. PhD Thesis, University of Twente.
Bonnema, G. M. (2012). *Thinking Tracks for Integrated Systems Design.* 1st Joint International Symposium on System-Integrated Intelligence 2012: New Challenges for Product and Production Engineering. Hannover, Germany.
Bonnema, G. M. (2014). Communication in multidisciplinary systems architecting. *In* 24th CIRP Design Conference, Milano, Italy.
Bonnema, G. M., Veenvliet, K. T. & Broenink, J. F. (2015). *Systems Design and Engineering: Facilitating Multidisciplinary Development Projects.* CRC Press; Enschede, The Netherlands.
Borches, P. D. (2010). *A3 Architecture overviews*. PhD Thesis, University of Twente.
Boucher, M. & Houlihan, D. (2008). System Design: New Product Development for Mechatronics. Tech.rep. Aberdeen Group;
Braa, K. & Vidgen, R. T. (1999). Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research. *Information and Organization,* vol. 9 (1)**,** pp. 25-47.
Braun, P., Broy, M., Houdek, F., Kirchmayr, M., Müller, M., Penzenstadler, B., Pohl, K. & Weyer, T. (2010). Guiding requirements engineering for software-intensive embedded systems in the automotive industry. *Computer Science - Research and Development,* vol. 29 (1)**,** pp. 21-43.
Broadfoot, G. H. (2005). ASD Case Notes: Costs and Benefits of Applying Formal Methods to Industrial Control Software. *In* International Symposium of Formal Methods Europe, Newcastle, UK.
Browning, T. R., Fricke, E. & Negele, H. (2006). Key concepts in modeling product development processes. *Systems Engineering,* vol. 9 (2)**,** pp. 104-128.
Brussel, F. (2014). *Interactive A3 Architecture Overviews - Intuitive Functionalities for Effective Communication (In Dutch)*. Bachelor Thesis, University of Twente.
Brussel, F. F. & Bonnema, G. M. (2015). Interactive A3 Architecture Overviews: Intuitive Functionalities for Effective Communication. *In* 2015 Conference on Systems Engineering Research (CSER), Hoboken, NY, USA.
Calvano, C. N. & John, P. (2004). Systems engineering in an age of complexity. *Systems Engineering,* vol. 7 (1)**,** pp. 25-34.
Canedo, A. & Richter, J. H. (2014). Architectural Design Space Exploration of Cyber-Physical Systems using the Functional Modeling Compiler. *In* 24th CIRP Design Conference, Milano, Italy.
Clark, H. H. (1996). *Using Language.* Cambridge University Press; New York, NY, USA.
Cloutier, R., Hamilton, D., Zigh, T., Korfiatis, P., Behnam Esfahbod, Zhang, P., Pape, P. & O'Brian, J. (2013). Graphical CONOPS prototype to demonstrate emerging methods, processes, and tools at ARDEC. Tech.rep. SERC-2013-TR-031-2. Stevens Institute of Technology & System Engineering Research Center; Hoboken, NY, USA.
COMMIT (2015). *About COMMIT* [Online]. Available: http://commit-nl.nl/about-commit [Accessed August 23rd 2015].
Controllab Products B.V. (2011). *Welcome to 20-sim, the software for modeling dynamic systems* [Online]. Available: http://www.20sim.com/index.html [Accessed September 11th 2011].
Conway, M. E. (1968). How do Committees Invent? *Datamation,* vol. 14 (5)**,** pp. 28-31.
Curry, M. D. & Ross, A. M. (2015). Considerations for an Extended Framework for Interactive Epoch-Era Analysis *In* 2015 Conference on Systems Engineering Research (CSER), Hoboken, NY, USA.
D'Ambrogio, A., Gianni, D., Fuchs, J. & Iazeolla, G. (2014). *Track Report of Collaborative Modeling and Simulation (CoMetS) track of WETICE 2014.* 2014 IEEE 23rd International Wetice Conference (WETICE). Parma, Italy.

Dell Corporation (2015). *Dell XPS 18* [Online]. Available: *http://www.dell.com/us/p/xps-18-1820-aio/pd* [Accessed August 23rd 2015].

Derler, P., Lee, E. A. & Vincentelli, A. S. (2012). Modeling cyber–physical systems. *Proceedings of the IEEE (special issue on CPS),* vol. 100 (1)**,** pp. 13-28.

Do, Q., Cook, S., Campbell, P., Scott, W., Robinson, K., Power, W. & Tramoundanis, D. (2012). Requirements for a Metamodel to Facilitate Knowledge Sharing between Project Stakeholders. *In* 2012 Conference on Systems Engineering Research (CSER), St. Louis, MO, USA.

Dori, D. (2002). *Object-Process Methodology – A Holistic Systems Paradigm.* Springer Verlag; New York, NY, USA.

Dori, D. (2011). Object-Process Methodology for Structure-Behavior Co-Design. *In:* Embley, D. W. & Thalheim, B. (eds.) *Handbook of Conceptual Modeling - Theory, Practice, and Research Challenges.* Springer, pp. 209-258.

Dori, D., Bolshchikov, S. & Wengrowicz, N. (2014). Conceptual models become alive with Vivid OPM: How can animated visualization render abstract ideas concrete? *In:* Gianni, D., D'Ambrogio, A. & Tolk, A. (eds.) *Modeling and Simulation-Based Systems Engineering Handbook.* CRC Press 2014, pp. 293-320.

Eclipse (2015). Eclipse Modeling Framework.

Embedded Systems Innovation by TNO (2011). *Allegio project* [Online]. Available: *http://www.esi.nl/allegio/* [Accessed September 11th 2012].

Embedded Systems Innovation by TNO (2014). *POOSL plugin for Eclipse* [Online]. Available: *http://poosl.esi.nl/download/* [Accessed August 23rd 2015].

Emery, D. & Hilliard, R. (2009). Every Architecture Description Needs a Framework: Expressing Architecture Frameworks Using ISO/IEC 42010. *In* 2009 Joint Working Ieee/Ifip Conference on Software Architecture and European Conference on Software Architecture, Cambridge, UK.

Erden, M. S., Komoto, H., van Beek, T. J., D'Amelio, V., Echavarria, E. & Tomiyama, T. (2008). A review of function modeling: approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* vol. 22 (2)**,** pp. 147-169.

ESA (2014). CDF Study Report - Phobos Sample Return - Phobos Moon of Mars Sample Return Mission [Online]. Available: *http://sci.esa.int/future-missions-office/55323-cdf-study-report-phobos-sample-return/*.

ESA (2015). *Concurrent Design Facility* [Online]. Available: *http://www.esa.int/Our_Activities/Space_Engineering_Technology/CDF* [Accessed August 23rd 2015].

Esfahani, N., Malek, S. & Razavi, K. (2013). GuideArch: guiding the exploration of architectural solution space under uncertainty. *In* Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA.

Estefan, J. A. (2008). Survey of Model-Based Systems Engineering (MBSE) Methodologies. Tech.rep. INCOSE-TD-2007-003-02. MBSE Initiative, INCOSE; Pasadena, CA, USA.

Figenbaum, E., Fearnley, N., Pfaffenbichler, P., Hjorthol, R., Kolbenstvedt, M., Jellinek, R., Emmerling, B., Bonnema, G. M., Ramjerdi, F., Vågane, L. & Iversen, L. M. (2015). Increasing the competitiveness of e-vehicles in Europe. *European Transport Research Review,* vol. 7 (3)**,** pp. 7-28.

Forsberg, K. J. & Mooz, H. (1991). The relationship of system engineering to the project cycle. *In* Proceedings of the First Annual Symposium of National Council on System Engineering, Chattanooga, TN, USA.

French, M. J. (1985). *Conceptual Design for Engineers.* 2nd ed. Springer; London, UK.

Freriks, H. J. M., Heemels, W. P. M. H., Muller, G. J. & Sandee, J. H. (2006). *On the Systematic Use of Budget-Based Design.* Proceedings of the 16th Annual International Symposium of the International Council on System Engineering (INCOSE2006). Orlando, FL, USA.

Garde, J. A. (2013). *Everyone has a part to play: games and participatory design in healthcare*. PhD Thesis, University of Twente.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine,* vol. 11 (4)**,** pp. 26-36.

Gero, J. S. & Kannengiesser, U. (2004). The situated function-behaviour-structure framework. *Design Studies,* vol. 25 (4)**,** pp. 373-391.

Gianni, D., D'Ambrogio, A. & Tolk, A. (eds.) (2014). *Modeling and Simulation-Based Systems Engineering Handbook.* CRC Press 2014;

Gries, M. (2003). Methods for Evaluating and Covering the Design Space During Early Design Development. Tech.rep. UCB/ERL M03/32. EECS Department, University of California; Berkeley, CA, USA.

Grogan, P. T. (2014). *Interoperable Simulation Gaming for Strategic Infrastructure Systems Design*. PhD Thesis, Massachusetts Institute of Technology.

Grogan, P. T. & Weck, O.L. de (2015). Interactive simulation games to assess federated satellite system concepts. *In* IEEE Aerospace Conference 2015, Big Sky, MT, USA.

Grogan, P. T., Weck, O.L. de, Ross, A. M. & Rhodes, D. H. (2015). Interactive Models as a System Design Tool: Applications to System Project Management. *2015 Conference on Systems Engineering Research (CSER),* vol. 44 (0)**,** pp. 285-294.

Gulati, R. K. & Eppinger, S. D. (1996). The coupling of product architecture and organizational structure decisions Tech.rep. #3906-96. Sloan School of Management; Massachusetts Institute of Technology; Cambridge, MA, USA.

Hansman, R. J., Magee, C., Neufville, R. D., Robins, R. & Roos, D. (2006). Research agenda for an integrated approach to infrastructure planning, design and management. *International Journal of Critical Infrastructures,* vol. 2 (2/3)**,** pp. 146.

Hansson, S. O. (1994). Decision Theory - A Brief Introduction. Tech.rep. Department of Philosophy and the History of Technology, Royal Institute of Technology (KTH); Stockholm, Sweden.

Haveman, S. P. (2009). *Project Buzz Tracker: Supporting System Architects in the Future Workspace*. Master Thesis, University of Twente.

Haveman, S. P. & Bonnema, G. M. (2013). Requirements for high level models supporting design space exploration in model-based systems engineering. *In* 2013 Conference on Systems Engineering Research (CSER), Atlanta, GA, USA.

Haveman, S. P. & Bonnema, G. M. (2015a). Communication of simulation and modelling activities in early systems engineering. *In* 2015 Conference on Systems Engineering Research (CSER), Hoboken, NY, USA.

Haveman, S. P. & Bonnema, G. M. (2015b). A conceptual model to support communication of systems modeling and simulation activities. *In* 9th Annual IEEE International Systems Conference (SysCon 2015), Vancouver, BC, Canada.

Haveman, S. P., Bonnema, G. M. & Berg, F.G.B. van den (2014). Early Insight in Systems Design through Modeling and Simulation. *In* 2014 Conference on Systems Engineering Research (CSER), Los Angeles, CA, USA.

Heemels, W. P. M. H. & Muller, G. (eds.) (2006). *Boderc: Model-based design of high-tech systems. A collaborative research project for multi-disciplinary design analysis of high-tech systems.* Embedded Systems Institute; Eindhoven, The Netherlands.

Heemels, W. P. M. H., Somers, L. J., van den Bosch, P., Yuan, Z., van der Wijst, B., van den Brand, A. & Muller, G. (2006a). *The Use of the Key Driver Technique in the Design of Copiers.* ICSSEA 2006. Paris, France.

Heemels, W. P. M. H., van de Waal, E. & Muller, G. (2006b). A multi-disciplinary and model-based design methodology for high-tech systems. *In* 2006 Conference on Systems Engineering Research (CSER), Los Angelese, CA, USA.

Heer, J. & Agrawala, M. (2007). Design Considerations for Collaborative Visual Analytics. *In* IEEE Symposium on Visual Analytics Science and Technology (VAST 2007), Sacramento, CA, USA.

Heer, J. & Shneiderman, B. (2012). Interactive Dynamics for Visual Analysis. *Queue,* vol. 10 (2)**,** pp. 30-55.

Hendriks, M., Basten, T., Verriet, J., Brassé, M. & Somers, L. (2014). A blueprint for system-level performance modeling of software-intensive embedded systems. *International Journal on Software Tools for Technology Transfer***,** pp. 1-20.

Hooman, J., Mooij, A. J. & Wezep, H. (2012). Early Fault Detection in Industry Using Models at Various Abstraction Levels. *In* 9th International Conference, IFM 2012, Pisa, Italy.

Houy, C., Fettke, P. & Loos, P. (2012). Understanding Understandability of Conceptual Models – What Are We Actually Talking about? *In* 31st International Conference on Conceptual Modeling, Florence, Italy.

Howard, R. (1966). Information Value Theory. *IEEE Transactions on Systems Science and Cybernetics,* vol. 2 (1)**,** pp. 22-26.

Husain, I. (2011). *Electric and Hybrid Vehicles: Design Fundamentals.* 2nd ed. CRC Press; Boca Raton, NY, USA.

IEEE (1990). 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. Institute of Electrical and Electronics Engineers; New York, NY, USA.

IEEE (2007). Std 1471-2000. Systems and software engineering - Recommended practice for architectural description of software-intensive systems. Institute of Electrical and Electronics Engineers; New York, NY, USA.

INCOSE (2013). Model-based Conceptual Design Working Group (MBCD WG) Charter [Online]. Available: *http://www.incose.org/docs/default-source/wgcharters/model-based-conceptual-design.pdf*.

Ionita, M. T. (2005). *Scenario-Based System Architecting*. PhD Thesis, Technical University of Eindhoven.

ISO/IEC/IEEE (2011). ISO/IEC/IEEE 42010 - Systems and software engineering: Architecture description Institute of Electrical and Electronics Engineers; New York, NY, USA.

Ivanovic, A. & America, P. (2010). Customer value in architecture decision making. *In* 4th European conference on Software architecture, Copenhagen, Denmark.

Jakobsson, Å. (2014). Application of MBCD to Creativity and Innovation. *Incose Insight,* vol. 17 (4)**,** pp. 33-34.

Kahn, H. & Mann, I. (1957). Ten Common Pitfalls. Tech.rep. RM-1937. The RAND Corporation; Santa Monica, CA, USA.

Kemble, S. (2006). *Interplanetary Mission Analysis and Design.* Springer; Berlin Heidelberg.

Kempen, M. J. C. (2015). *Improving the customers' experience of Vanderlande Baggage Handling Systems* Master Thesis, University of Twente.

Keshishzadeh, S. (2016). *(Forthcoming) Formal Analysis and Verification of Embedded Systems for Healthcare*. PhD Thesis, Technical University Of Eindhoven.

Kienhuis, B., Deprettere, E., Vissers, K. & Wolf, P. v. d. (1997). An approach for quantitative analysis of application-specic dataflow architectures. *In* IEEE International Conference on Application-Specic Systems, Architectures, and Processors (ASAP 1997), Zurich, Switzerland.

Klein, P., Lützenberger, J., Kristensen, K., Iversen, G. & Pawar, K. (2014). K-Brief and Context Driven Access - Providing context related information to product developers in high quality. *In* 2014 International ICE Conference on Engineering, Technology and Innovation (ICE), Bergamo, Italy.

Knight, M. & Vencel, L. (2014). Problem Framing: How Can Model-Based Methods Help Systems Engineers Solve The Right Problem? *Incose Insight,* vol. 17 (4)**,** pp. 26-28.

Kolfschoten, G., Lukosch, S. & Mathijssen, A. (2012). Supporting Collaborative Design: Lessons from a case study at the ESA concurrent design facility. *In* International Conference on Group Decision and Negotiation (GDN), Recife, Brazil.

Koning, H. P. de, Gerené, S., Ferreira, I., Pickering, A., Beyer, F. & Vennekens, J. (2014). *Open Concurrent Design Tool - ESA Community Open Source - Ready to Go.* SECESA 2014 Stuttgart, Germany.

Kooistra, R. L., Kamp, P. G. & Bonnema, G. M. (2014). *The cause of complications: understanding the relation between post-operative complications and the systems and processes of a hospital by means of an influence diagram.* 23rd Congress of the International Federation of Hospital Engineering (IFHE). Buenos Aires, Argentina.

Korfiatis, P., Zigh, T. & Blackburn, M. (2012). Graphical CONOPS Development to Enhance Model Based Systems Engineering. *In* 2012 Industrial and Systems Engineering Research Conference (IIE ISERC Annual Conference), Orlando, FL, USA.

Koziolek, A. (2012). Architecture-driven quality requirements prioritization. *In* First IEEE International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), Chicago, IL, USA.

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE Software,* vol. 12 (6)**,** pp. 42-50.

Kuuluvainen, I., Vanttinen, M. & Koskinen, P. (1991). The Action-State Diagram: A Compact Finite State Machine Representation For User Interfaces And Small Embedded Reactive Systems. *IEEE Transactions on Consumer Electronics,* vol. 37 (3).

Lapouchnian, A. (2005). Goal-Oriented Requirements Engineering: An Overview of the Current Research. Tech.rep. Department of Computer Science, University of Toronto; Toronto, ON, Canada.

Law, A. M. (2014). *Simulation Modeling and Analysis.* 5th ed. McGraw-Hill;

Lee, B. D. & Paredis, C. J. J. (2014). A Conceptual Framework for Value-driven Design and Systems Engineering. *In* 24th CIRP Design Conference, Milan, Italy.

Liu, Z. L., Zhang, Z. & Chen, Y. (2012). A scenario-based approach for requirements management in engineering design. *Concurrent Engineering,* vol. 20 (2)**,** pp. 99-109.

Lockheed Martin (2015). *Skunk Works - Defining The Future Of Aerospace* [Online]. Available: *http://www.lockheedmartin.com/us/aeronautics/skunkworks.html* [Accessed September 9th 2015].

Lu, S. C. Y., Elmaraghy, W., Schuh, G. & Wilhelm, R. (2007). A Scientific Foundation Of Collaborative Engineering. *CIRP Annals - Manufacturing Technology,* vol. 56 (2)**,** pp. 605-634.

Lutters, D. (2001). *Manufacturing integration based on information management*. PhD Thesis, University of Twente.

Madni, A. M., Nance, M., Richey, M., Hubbard, W. & Hanneman, L. (2014). Toward an Experiential Design Language: Augmenting Model-based Systems Engineering with Technical Storytelling in Virtual Worlds. *In* 2014 Conference on Systems Engineering Research (CSER 2014), Los Angeles, CA, USA.

Maier, M. & Rechtin, E. (2000). *The Art of Systems Architecting.* 2nd ed. CRC Press;

Malak, R. J., Jr, Aughenbaugh, J. M. & Paredis, C. J. J. (2009). Multi-attribute utility analysis in set-based conceptual design. *Computer Aided Design,* vol. 41 (3)**,** pp. 214-227.

March, J. G. (1978). Bounded Rationality, Ambiguity, and the Engineering of Choice. *Bell Journal of Economics,* vol. 9 (2)**,** pp. 587-608.

Mark, G. (2002). Extreme collaboration. *Communications of the ACM,* vol. 45 (6)**,** pp. 89-93.

Martinet, Y. (2014). *Lean proved to be efficient in manufacturing. Now, what about Lean in Early-Phase Systems Engineering? .* SECESA 2014 Conference. Stuttgart, Germany.

Mavris, D. (2009). Simulation Driven System of Systems Engineering Methods (Presentation). Tech.rep. National Defense Industrial Association, Modeling & Simulation Committee;

Mavris, D. N., Biltgen, P. T. & Weston, N. R. (2005). *Advanced Design of Complex Systems Using the Collaborative Visualization Environment (CoVE)* 43rd AIAA Aerospace Sciences Meeting and Exhibit. Reno, NV, USA.

Melching, M. (2012). *The System Design Communications Tool: supporting interactive A3 Architecture Overviews*. Master Thesis, University of Twente.

Microsoft (2010a). Excel 2010.

Microsoft (2010b). Visio 2010.

Mizuno, S. & Akao, Y. (1994). *QFD: The Customer-driven Approach to Quality Planning & Deployment.* Asian Productivity Organization; Tokyo, Japan.

Modelica Association (2012). *Modelica and the Modelica Association* [Online]. Available: *https://modelica.org/* [Accessed September 10th 2012].

Moneva, H., Hamberg, R. & Punter, T. (2011). *A Design Framework for Model-based Development of Complex Systems.* 32nd IEEE Real-Time Systems Symposium, 2nd Analytical Virtual Integration of Cyber-Physical Systems Workshop. Vienna, Austria.

Mooij, A. J., Hooman, J. & Albers, R. (2013). Early Fault Detection using Design Models for Collision Prevention in Medical Equipment. *In* Foundations of Health Information Engineering and Systems (FHIES 2013), Macau, China.

Morris, B. A., Harvey, D. & Do, Q. (2015). *Survey of Model-Based Conceptual Design Challenges.* SETE 2015. Canberra, Australia.

Morse, K. L., Coolahan, J., Lutz, B., Horner, N. C., Vick, S. & Syring, R. (2010). Best Practices For The Development Of Models And Simulations. Tech.rep. NSAD-R-2010-037. Johns Hopkins University - Applied Physics Laboratory Laurel, MD, USA.

Muller, G. J. (2004). *CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity*. PhD Thesis, Technical University of Delft.

Muller, G. J. (2009). Systems Engineering Research Validation [Online]. Available: *http://www.gaudisite.nl/SEresearchValidationPaper.pdf*.

Muller, G. J. (2010). Industry-as-Laboratory Applied in Practice: The Boderc Project [Online]. Available: *http://www.gaudisite.nl/IndustryAsLaboratoryAppliedPaper.pdf*.

Muller, G. J. (2011a). System Architecting [Online]. Available: *http://www.gaudisite.nl/SystemArchitectureBook.pdf*.

Muller, G. J. (2011b). *System Architecting: A Business Perspective.* CRC Press;

Muller, G. J. (2013). Systems Engineering Research Methods. *In* 2013 Conference on Systems Engineering Research (CSER), Atlanta, GA, USA.

Muller, G. J. (2014). Consolidating Architecture Overviews [Online]. Available: *http://www.gaudisite.nl/info/ConsolidatingArchitectureOverviews.info.html*.

NASA (2008). NASA-STD-7009. National Aeronautics and Space Administration; Washington, DC, USA.

National Research Council (2002). Modeling and Simulation in Manufacturing and Defense Acquisition: Pathways to Succes. Tech.rep. Washington, DC, USA.

Neumann, J. von & Morgenstern, O. (1944). *Theory of games and economic behavior* Princeton University Press; Princeton, NJ, USA.

Nonaka, I. & Takeuchi, H. (1995). *The Knowledge-Creating Company.* Oxford University Press New York, NY, USA.

Noppen, J., Broek, P. v. d. & Akit, M. (2007). Software development with imperfect information. *Soft Comput.,* vol. 12 (1)**,** pp. 3-28.

Object Management Group Inc. (2012). *OMG SysML* [Online]. Available: *http://www.omgsysml.org/#What-Is_SysML* [Accessed September 10th 2012].

OECD (2010). Health care systems: Getting more value for money [Online]. Available: *http://www.oecd.org/eco/growth/46508904.pdf*.

Oracle Corporation (2015). *JAVA* [Online]. Available: *https://www.oracle.com/java/index.html* [Accessed August 23rd 2015].

Osburg, J. & Mavris, D. N. (2005). A Collaborative Design Environment to Support Multidisciplinary Conceptual Systems Design. *In* SAE World Aerospace Congress 2005, Dallas, TX, USA.

Ouwerkerk, H. & Canjels, I. (2011). A3 Architecture Overview of System Latency. Tech.rep. Philips Interventional X-ray Division (Internal Document); Best, The Netherlands.

Pahl, G., Beitz, W., Feldhusen, J. & Grote, K.-H. (2007). *Engineering Design - A Systematic Approach.* 3rd English ed. Springer;

Potts, C. (1993). Software-Engineering Research Revisited. *IEEE Software,* vol. 10 (5)**,** pp. 19-28.

Putten, P.H.A. van der & Voeten, J. P. M. (1997). *Specification of Reactive Hardware/Software Systems - The method Software/Hardware Engineering (SHE)*. PhD Thesis, Technical University of Eindhoven.

Rajabalinejad, M. & Bonnema, G. M. (2014). Determination of stakeholders' consensus over values of system of systems. *In* System of Systems Engineering (SOSE), 2014 9th International Conference on, Adelaide, Australia.

Reichwein, A. & Paredis, C. J. J. (2011). Overview of Architecture Frameworks and Modeling Languages for Model-Based Systems Engineering. *In* ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Washington, DC, USA.

Ricci, N., Schaffner, M. A., Ross, A. M., Rhodes, D. H. & Fitzgerald, M. E. (2014). Exploring Stakeholder Value Models via Interactive Visualization. *In* 2014 Conference on Systems Engineering Research (CSER), Los Angeles, CA.

Robinson, K., Waite, M. & Do, Q. (2014). Introduction to the Model-Based Conceptual Design Special Issue. *Incose Insight,* vol. 17 (4)**,** pp. 7.

Robinson, S. (2008a). Conceptual Modelling for Simulation Part I: Definition and Requirements. *Journal of the Operational Research Society,* vol. 59 (3)**,** pp. 278-290.

Robinson, S. (2008b). Conceptual Modelling for Simulation Part II: A Framework for Conceptual Modelling. *Journal of the Operational Research Society,* vol. 59 (3)**,** pp. 291-304.

Ross, A. M. (2006). *Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration*. PhD Thesis, Massachusetts Institute of Technology.

Royal Philips (2015a). *Allura Xper FD20* [Online]. Available: http://www.healthcare.philips.com/main/products/interventional_xray/product/interventional_vascular_surgery/imaging_systems/vascsurgery_fd20.wpd [Accessed August 23rd 2015].

Royal Philips (2015b). *Philips Interventional X-ray* [Online]. Available: http://www.healthcare.philips.com/main/products/interventional_xray/Product/ [Accessed August 23rd 2015].

Royce, W. W. (1970). Managing the Development of Large Software Systems. *In* IEEE WESCON, Los Angeles, CA, USA.

Rozanski, N. & Woods, E. (2012). *Software Systems Architecture.* Addison Wesley Upper Saddle River.

Ryan, J., Sarkani, S. & Mazzuchi, T. (2014). Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis. *Systems Engineering,* vol. 17 (1)**,** pp. 10-25.

Sampson, M. & Friedenthal, S. (2014). Role of MBCD in the INCOSE MBSE Initiative. *Incose Insight,* vol. 17 (4)**,** pp. 10.

Sargent, R. G. (2013). Verification and validation of simulation models. *Journal of Simulation,* vol. 7 (1)**,** pp. 12-24.

Schramm, W. (1954). *The Process and Effects of Mass Communication.* University of Illinois Press; Urbana, IL, USA.

Schuitemaker, K., Braakhuis, J. G. & Rajabalinejad, M. (2015). *A model based safety architecture framework for Dutch high speed train lines.* 10th International Conference on System of Systems Engineering (SoSE). San Antonio, TX, USA.

Schumann, H., Wendel, H., Braukhane, A., Berres, A., Gerndt, A. & Schreiber, A. (2010). Concurrent Systems Engineering in Aerospace: From Excel-based to Model Driven Design. *In* 2010 Conference on Systems Engineering Research (CSER), Hoboken, NJ, USA.

Schuts, M. & Hooman, J. (2015a). Formalizing the Concept Phase of Product Development. *In* Formal Methods - 20th International Symposium, Oslo, Norway.

Schuts, M. & Hooman, J. (2015b). Using Domain Specific Languages to Improve the Development of a Power Control Unit. *In* 2015 Federated Conference on Computer Science and Information Systems, Lodz, Poland.

Shah, A. A., Kerzhner, A. A., Schaefer, D. & Paredis, C. J. J. (2010). Multi-view modeling to support embedded systems engineering in SysML. *In:* Engels, G., Lewerentz, C., Schäfer, W., Schürr, A. & Westfechtel, B. (eds.) *Graph Transformations and Model-Driven Engineering.* Springer, pp. 580-601.

Shannon, C. & Weaver, W. (1949). *The Mathematical Theory of Communication.* University of Illinois Press; Urbana, IL, USA.

Sikora, E., Tenbergen, B. & Pohl, K. (2011). Requirements Engineering for Embedded Systems - An Investigation of Industry Needs. *In* 17th International Working Conference, REFSQ 2011, Essen, Germany.

Singh, V. & Muller, G. (2013). *Knowledge Capture, Cross Boundary Communication and Early Validation with Dynamic A3 Architectures.* INCOSE International Symposium (IS 2013). Philadelphia, PA, USA.

Sobek, D. K., Ward, A. C. & Liker, J. K. (1999). Toyota's principles of set-based concurrent engineering. *Sloan Management Review,* vol. 40 (2)**,** pp. 67-83.

Sohlenius, G. (1992). Concurrent Engineering. *CIRP Annals - Manufacturing Technology,* vol. 41 (2)**,** pp. 645-655.

Suh, N. P. (1990). *The Principles of Design.* Oxford University Press;

Suh, N. P. (2005). Complexity in Engineering. *CIRP Annals - Manufacturing Technology,* vol. 54 (2)**,** pp. 46-63.

Taleb, N. N. (2007). *The Black Swan: The Impact of the Highly Improbable.* Random House; New York, NY, USA.

Tesla Motors (2015). *Model S | Tesla Motors* [Online]. Available: *http://www.teslamotors.com/models* [Accessed August 23rd 2015].

Thalen, J. (2014). *Facilitating User Centered Design Through Virtual Reality*. PhD Thesis, University of Twente.

Theelen, B. D., Florescu, O., Geilen, M. C. W., Huang, J., van der Putten, P. H. A. & Voeten, J. P. M. (2007). Software/Hardware Engineering with the Parallel Object-Oriented Specification Language. *In* 5th ACM & IEEE International Conference on Formal Methods and Models for Co-Design, Nice, France.

Theelen, B. D. & Hooman, J. (2015). Uniting Academic Achievements on Performance Analysis with Industrial Needs. *In* 12th International Conference QEST 2015, Madrid, Spain.

Thompson, S. C. & Paredis, C. J. J. (2010). An Investigation Into the Decision Analysis of Design Process Decisions. *Journal of Mechanical Design,* vol. 132 (12).

Tomiyama, T. (1994). From general design theory to knowledge-intensive engineering. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing,* vol. 8 (4)**,** pp. 319-333.

Tomiyama, T., D'Amelio, V., Urbanic, J. & ElMaraghy, W. (2007). Complexity of Multi-Disciplinary Design. *CIRP Annals - Manufacturing Technology,* vol. 56 (1)**,** pp. 185-188.

Tomiyama, T., Gu, P., Jin, Y., Lutters, D., Kind, C. & Kimura, F. (2009). Design methodologies: Industrial and educational applications. *CIRP Annals - Manufacturing Technology,* vol. 58 (2)**,** pp. 543-565.

Tomiyama, T. & Yoshikawa, H. (1986). Extended general design theory. Tech.rep. CS-R8604. Centre for Mathematics and Computer Science; Amsterdam, The Netherlands.

Topper, J. S. & Horner, N. C. (2013). Model-Based Systems Engineering in Support of Complex Systems Development. *Johns Hopkins Applied Physics Laboratory Technical Digest,* vol. 32 (1)**,** pp. 419-432.

Torry-Smith, J. M., Qamar, A., Achiche, S., Wikander, J., Mortensen, N. H. & During, C. (2011). Mechatronic Design: Still A Considerable Challenge. *In* 23rd International Conference on Design Theory and Methodology (DTM), Washington, DC, USA.

Tsiolkovsky, K. (1903). The Exploration of Cosmic Space by Means of Reaction Devices (Исследование мировых пространств реактивными приборами). *The Science Review (in Russian)*.

Tversky, A. & Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. *Science, New Series,* vol. 185 (4157)**,** pp. 1124-1131.

U.S. Department of Defense (2010). *DoDAF - U.S. Department of Defense Architecture Framework v2.0* [Online]. Available: *http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf* [Accessed October 27th 2014].

U.S. Department of Transportation (2007). Systems Engineering for Intelligent Transportation Systems [Online]. Available: *http://ops.fhwa.dot.gov/publications/seitsguide/*.

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. & Tomiyama, T. (1996). Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing,* vol. 10 (4)**,** pp. 275-288.

Umeda, Y., Takeda, H., Tomiyama, T. & Yoshikawa, H. (1990). Function, behaviour, and structure. *Applications of artificial intelligence in engineering,* vol. 1**,** pp. 177-194.

Valente, A. & Marchetti, E. (2010). Development of a Rich Picture editor: a user-centered approach. *International Journal on Advances in Intelligent Systems,* vol. 3 (3 & 4)**,** pp. 187-199.

Walden, D. D., Roedler, G. J., Forsberg, K. J., Hamelin, R. D. & Shortell, T. M. (eds.) (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities.* INCOSE; San Diego, CA, USA.

Wang, R. & Dagli, C. H. (2011). Executable system architecting using systems modeling language in conjunction with colored Petri nets in a model-driven systems development process. *Systems Engineering,* vol. 14 (4)**,** pp. 383-409.

Weck, O.L. de, Eckert, C. & Clarkson, J. (2007). *A classification of uncertainty for early product and system design.* ICED '07 - International Conference On Engineering Design. Paris, France.

Wikipedia (2015a). *Angiography* [Online]. Available: *https://en.wikipedia.org/wiki/Angiography* [Accessed August 23rd 2015].

Wikipedia (2015b). *X-Ray* [Online]. Available: *https://en.wikipedia.org/wiki/X-ray* [Accessed August 23rd 2015].

Woestenenk, K. (2014). *Consistency, integration, and reuse in multi-disciplinary design processes*. PhD Thesis, University of Twente.

Xu, T., Hutfless, S. M., Cooper, M. A., Zhou, M., Massie, A. B. & Makary, M. A. (2015). Hospital cost implications of increased use of minimally invasive surgery. *JAMA Surgery,* vol. 150 (5)**,** pp. 489-490.

Yaroker, Y., Perelman, V. & Dori, D. (2013). An OPM conceptual model-based executable simulation environment: Implementation and evaluation. *Systems Engineering,* vol. 16 (4)**,** pp. 381-390.

Zachman, J. (2008). John Zachman's Concise Definition of the Zachman Framework [Online]. Available: *http://www.zachman.com/about-the-zachman-framework*.

Zeigler, B. P., Praehofer, H. & Kim, T. G. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* 2nd ed. Academic Press; New York, NY, USA.

About the Author

Steven Pieter Haveman was born on August 1st, 1986 in Voorburg. Growing up in Zoetermeer, he returned to Voorburg for secondary school. Here, he completed his Gymnasium, specializing in Nature & Technology, but with the inclusion of Economics and Art. He continued this combination in his Bachelor Industrial Design at the University of Twente. During his Bachelor, which he completed in 2007, he became more interested in designing systems. Therefore, he chose "Smart Products and Environments" as track in his Master Industrial Design Engineering, again at the University of Twente. Steven graduated in 2009 on a support tool for System Architects in the future workplace. This tool, "Project Buzz Tracker", aims to help System Architects uncover, filter and select relevant architecting information.

After graduation, Steven worked as junior project manager for the European Design Centre. Here he was involved in new project developments, proposal writing, (technical) market research and visualizations of systems and information. These activities focused on various areas such as serious gaming (both physical and virtual), visualization technologies and supportive activities for creative industries. In November 2011, Steven returned to the University of Twente to start his PhD research. This research has been conducted in close collaboration with Philips Healthcare, where Steven has been embedded as a researcher for the duration of the project.

Next to his professional activities, Steven is an active (beach) volleyball player and sports fan in general. Steven has also trained and coached various volleyball teams. Other pastimes are cycling, chess, computer games and lately, (wildlife) photography.

The following publications are part of the research presented in this thesis.

1. Haveman, S. P. & Bonnema, G. M. (2015). A conceptual model to support communication of systems modeling and simulation activities. *In* 9th Annual IEEE International Systems Conference (SysCon 2015), Vancouver, BC, Canada.
2. Haveman, S. P. & Bonnema, G. M. (2015). Communication of simulation and modelling activities in early systems engineering. *In* 2015 Conference on Systems Engineering Research (CSER), Hoboken, NY, USA.
3. Haveman, S. P., Bonnema, G. M. & Berg, F.G.B. van den (2014). Early Insight in Systems Design through Modeling and Simulation. *In* 2014 Conference on Systems Engineering Research (CSER), Los Angeles, CA, USA.
4. Haveman, S. P. & Bonnema, G. M. (2013). Requirements for high level models supporting design space exploration in model-based systems engineering. *In* 2013 Conference on Systems Engineering Research (CSER), Atlanta, GA, USA.

Appendix A Industry Interview Summaries

This section summarizes the various interviews that were conducted in industry to support the analysis of the state of the practice in Chapter 3 and Chapter 4. All the interview results were processed anonymously. Therefore, only the role of the interviewee and the company is listed. Furthermore, a short summary of the interview is given. The appendix is divided in interviews held at Philips iXR and interviews held at other companies.

A.1 Philips iXR Interviews

Interview 1			
Role of Interviewee	Value Engineer	**Date**	13-4-2012 and 3-5-2012
Organization & Domain	Philips iXR - Medical	**Type**	Face-to-Face Interview
Main Discussion Points	Role responsibilities, Tools & Methods Used		
The main task of a value engineer is developing new system solutions with an adapted value that can address specific customers or market segments. New solutions often focus on reduction of cost price, but focus on maintaining functionality at an acceptable level. A toolbox of nine methods is used, depending on the situation. The toolbox is a personal approach and is not implemented companywide despite efforts to do so. Toolbox does not cover extensive look at end user wishes for the system. A fixed toolset is too rigid or the tool 'explodes, like for example experiences with QFD. The tools are grouped into three categories; basic cost reduction, functional specifications and performance specifications. Examples are Design For Manufacture and Assembly, which concerns a Bill of Material analysis or Functional Value Analysis, which identifies value and role of functions.			

Interview 2			
Role of Interviewee	Requirements Engineer	**Date**	25-7-2012
Organization & Domain	Philips iXR - Medical	**Type**	Face-to-Face Interview
Main Discussion Points	Requirements: software, consistency and during design		
The use of requirements management software was recently started. Currently design is still document based. Several design documents have been converted to the software. The software can export documents, which can be used in the document based quality reviews again. At iXR, there is always more than one version of the product in the field or in development. This means there are also more than one set of requirements at the same time. Backwards compatibility is aimed for as much as possible. Limitation of the current software is that it does not support branching of requirements. Currently requirements are defined, decomposed and reviewed on testability by testers and on content by stakeholders. This process repeats itself on lower levels. Requirements traceability matrices are used to identify which links have been fulfilled between different requirements levels.			

Interview 3			
Role of Interviewee	System Architect	**Date**	27-7-2012
Organization & Domain	Philips iXR - Medical	**Type**	Face-to-Face Interview
Main Discussion Points	User Needs, Validation Strategies		
User Needs used to be part of main requirements document. However, it was decided to extract them from this document for increased visibility. This focuses mainly on the clinical end-user. Within Philips iXR validation happens on three levels. Product validation checks if the system is designed according to specifications. System operation is tested with prototypes and test builds. User			

satisfaction is tested with use tests. The user needs specification helps the validation of the last point. The level of detail of a need is coupled to the maturity of the need. More mature means that the requirements are more detailed and fixed. It is meant to be a multidisciplinary document
Some other quotes were that *"MBSE is often too rigid and requires too much. A lighter MBSE approach is hard to attain, but exceedingly valuable"* and that if a designer has more insight in context this is a good thing but if a designer considers himself as a user, many things get overlooked.

Interview 4			
Role of Interviewee	System Architect	**Date**	6-2-2013
Organization & Domain	Philips iXR - Medical	**Type**	Face-to-Face Interview
Main Discussion Points	Innovation Process, New System Development Strategies		
The technical issues that are relevant in new developments have a tension with the involved group of stakeholders. Lots of work goes into stakeholder alignment, as many different viewpoints have to be accounted for. Generally, innovation starts with a study, which can be continued into a technical concept, which can continue into concept release. A concept release is described with a project plan which can be committed to. An implementation project is started. Larger studies are done by the Philips Innovation division, smaller studies within the R&D department. Study is not always necessary; talking with stakeholders can help a lot as well. Innovative ideas go into funnels, different types of funnels exist such as feature, reliability and cost-down. During design, documentation is important for regulatory purposes (FDA). However, the goal is to have less documentation on the subsystem or component level.			

Interview 5			
Role of Interviewee	Product Architect (Hardware)	**Date**	6-2-2013
Organization & Domain	Philips iXR – Medical	**Type**	Face-to-Face Interview
Main Discussion Points	System Hardware, New Developments		
New hardware developments are often done with trial and error through testing with new hardware elements. It is aimed to maintain performance after hardware changes. Software architecture is driving hardware requirements. Innovation department does architecture type studies, R&D department does studies were solution direction is known. Drivers for development are cost price, reliability and functionality. To do successful modeling time, input and space (figuratively) is needed. Furthermore, critical remarks were made regarding the requirements process where often requirements are missing or underspecified, wrongly understood, have no clear goal and it is not possible to guarantee coverage 100%. Furthermore, stakeholders have a lack of insight in the system, which becomes apparent during design reviews. Not enough time & attention might be a reason. Furthermore, absence of stakeholders is sometimes an issue.			

Interview 6			
Role of Interviewee	Software Designer	**Date**	7-2-2013
Organization & Domain	Philips iXR - Medical	**Type**	Face-to-Face Interview
Main Discussion Points	Software, New Developments		
As software designer, often at end of new development pipeline. Involved in studies that aim to reduce risk of new development to acceptable level. In software, trend is visible of moving functionality from analogue implementations to pure software applications on general purpose computing platforms. The platform often dictates the process. Often mindset is functionality before performance, but performance could be considered earlier. Blockades are often worked around, easier than removing.			

A.2 Other Interviews

Interview 7			
Role of Interviewee	System Architect	**Date**	7-5-2014
Organization & Domain	FEI - Optics	**Type**	Face-to-Face Interview
Main Discussion Points	Modeling and Simulation, Innovations		
The need for modeling and simulation was discussed in various areas of the company. It was determined that several aspects of the system, while complex in functionality do not benefit from performance modeling as the performance requirements are not drivers of the design process. This is for example in the imaging pipeline. The accuracy of the optics is a major performance driver and has issues embedded in multiple disciplines. Here modeling could be worthwhile. Furthermore, the calibration process of the optics is a very complex process that both takes a long time and is very sensitive. Modeling this process would focus on process and manufacturing modeling.			

Interview 8			
Role of Interviewee	Line Manager	**Date**	4-7-2014
Organization & Domain	VanDerLande - Logistics	**Type**	Face-to-Face Interview
Main Discussion Points	Modeling and Simulation, Development Process		
Modeling and simulation is used a lot at VanDerLande. About 80% of the insight can be gained through this process. Emulations are used for the remainder of the process. At VanDerLande, systems are sold by sales engineers, after that a project manager turns up. The transfer between them is done with an open-up, which identifies for example the organization and relevant individuals in the client organization and a transfer of drawings. VanDerLande uses seven layers of abstraction in design, from the system level until the hardware component level.			

Interview 9			
Role of Interviewee	Functional Specialist	**Date**	3-12-2014
Organization & Domain	Airbus Defense & Space Netherlands - Space	**Type**	Face-to-Face Interview
Main Discussion Points	Use of Modeling & Simulation Tools, Development Projects		
Projects are led by a project manager and a system engineer who involve domain specialists. Larger projects include multiple specialists of the same domain. Airbus D&S NL does a lot of contracting work for ESA. There are strict standards for development projects. The process follows the ECSS standard. Projects included the European Robot Arm. This project has run exceptionally long because of launch vehicle problems. Design keeps being updated due to new launch requirements, continued guarantee of operation lifetime or new functionality requests A challenge is the systems engineering process in totality. Another challenge is that there is often a solution focused mindset which leads to missed opportunities due to lack of exploration. A further challenge is the transfer of results to other stakeholders, it is important to select right parameters for communication. System engineers play a big role in this. However, often time pressure forces one to communicate right away if you for example get pulled into a meeting. Each domain specialist has his own tool(set). No fixed guidelines on documentation of analysis results. Simulation is an art – to determine how much effort you will spend. Communication using A3AO's reminds interviewee of astronaut instruction sheets. These are intelligent people with limited time wanting to get a quick overview of a system. Finally, ESA ESTEC collaborative engineering was mentioned.			

Interview 10			
Role of Interviewees	Simulation Specialist (2x)	**Date**	11-5-2015
Organization & Domain	ESA - Space	**Type**	Phone Interview
Main Discussion Points	Concurrent Design Facility (CDF), Use of Simulations and Models		

Note: this was an interview with two interviewees simultaneously
Simulations in CDF are aimed towards trade-off decisions. Main analysis in CDF is based on structural analysis; no failure analysis is done in-depth. Simulations that include multiple disciplines are not done, as this is too complex. Process relies heavily on experts that are experienced in CDF process and are good in estimations. CDF acts mainly as budget and risk analysis, its goal is to be quick, but decently reliable.
The CDF process is a spiral process, each session a baseline is refined. Specialist teams work separately, but often share results in presentations. They are colocated, so they can and will also share intermediary updates in 1:1 conversations. However, people are also present via video or tele conferencing. There is no multidisciplinary overview of information. There is however an underlying MBSE database, the OCDT tool.
Most challenging aspect is to define the problem properly. Last improvement was the OCDT which helps subsystems to drive the design better.

Interview 11			
Role of Interviewee	Systems Engineer	**Date**	2-6-2015
Organization & Domain	ESA - Space	**Type**	Phone Interview
Main Discussion Points	Systems Engineering in CDF		

The SE view in the CDF mainly focuses on mass and electrical power, as those are the key drivers of most developments. This is a budget based approach. The physical view is used in design discussions on physical properties of the system.
The system engineer role and team leader (project manager) role are sometimes intertwined or even executed by one person. Interviewee has a specific background; there the focus is more on other areas, as these are more unknown to him.
OCDT tool does not update system design, only pushes out a parameter update. Subsystem designs still need to be adapted. The architecture of a system in the CDF is partly determined before the process starts through the selection of experts. E.g. if no nuclear propulsion is expected, that specialist is not present.
Domain specialists have ingrained that changing parameters or budgets impacts the rest of the system, they will be careful. CDF uses mostly high reliability components to compose the design. The design is used for a next stage, which cannot be executed at ESA due to how ESA functions. This will be a contract stage. What-if situations are accounted for during design by the risk specialist. Risk engineer follows design discussions and assesses aspects.
Simulations are not often used, but the system is considered across states. Each state is discussed separately.
Important issues are that communication is appropriate; the OCDT must be up to date. If the process is followed correctly, this will be more or less ensured.

Appendix B Hand-Eye Coordination Case Study Details

This appendix provides further detail for the hand-eye coordination case study described in section 5.2.

B.1 High Level Model User Interface

This section shows various screenshots of the Excel support tool in Figure 5.10.

- The *configurations overview* in Figure B.1 is the home screen of the tool.
- The *system configuration screen* in Figure B.2 allows definitions and editing of system models. In Figure B.2 the detector component in the platform view is selected. Therefore, the details of this component can be edited in the top-right part of the screen.
- The *simulation set-up screen* in Figure B.3 shows that it is possible to duplicate a single simulation run and each simulation run can simulate a user defined number of images
- The *results overview* in Figure B.4 gives an overview of basic results for all simulation runs done in a single batch of simulation runs
- The *detailed results screen* in Figure B.5 shows results of a single simulation run in detail. Various metrics are detailed in the top left. It is possible to "zoom" in, on the lower graph by specifying a different start and end image.

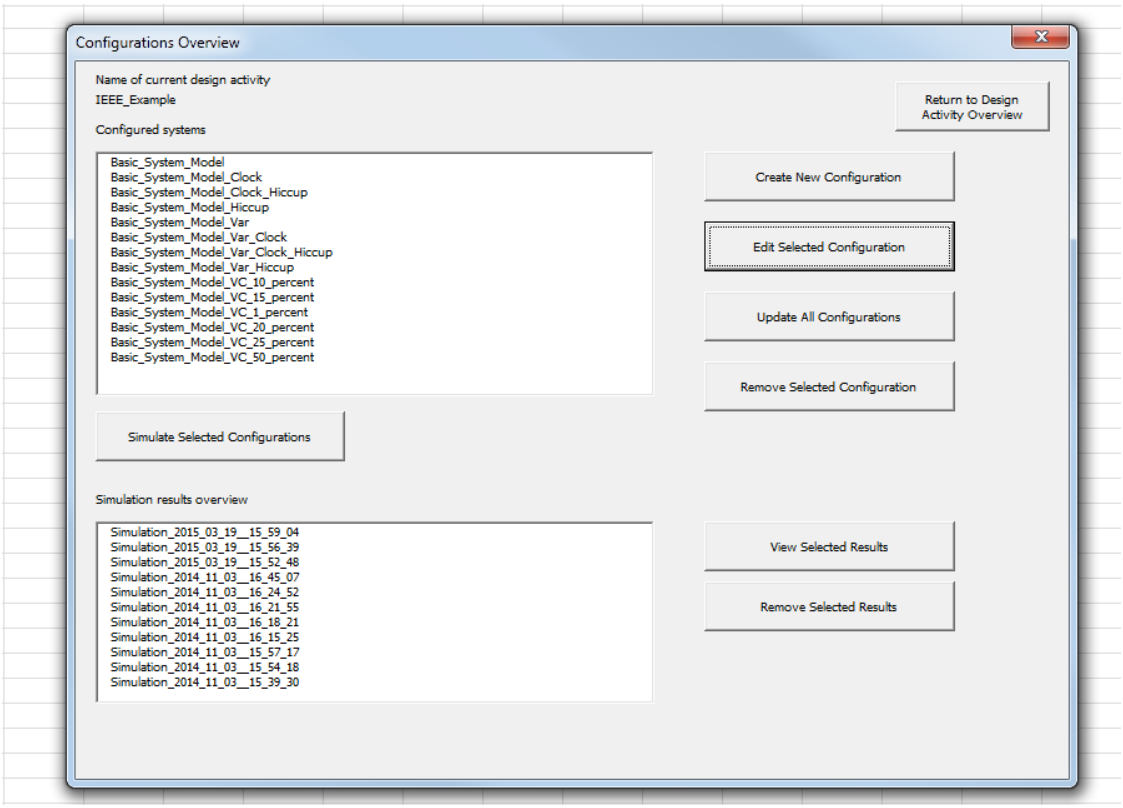


Figure B.1 – Configurations Overview

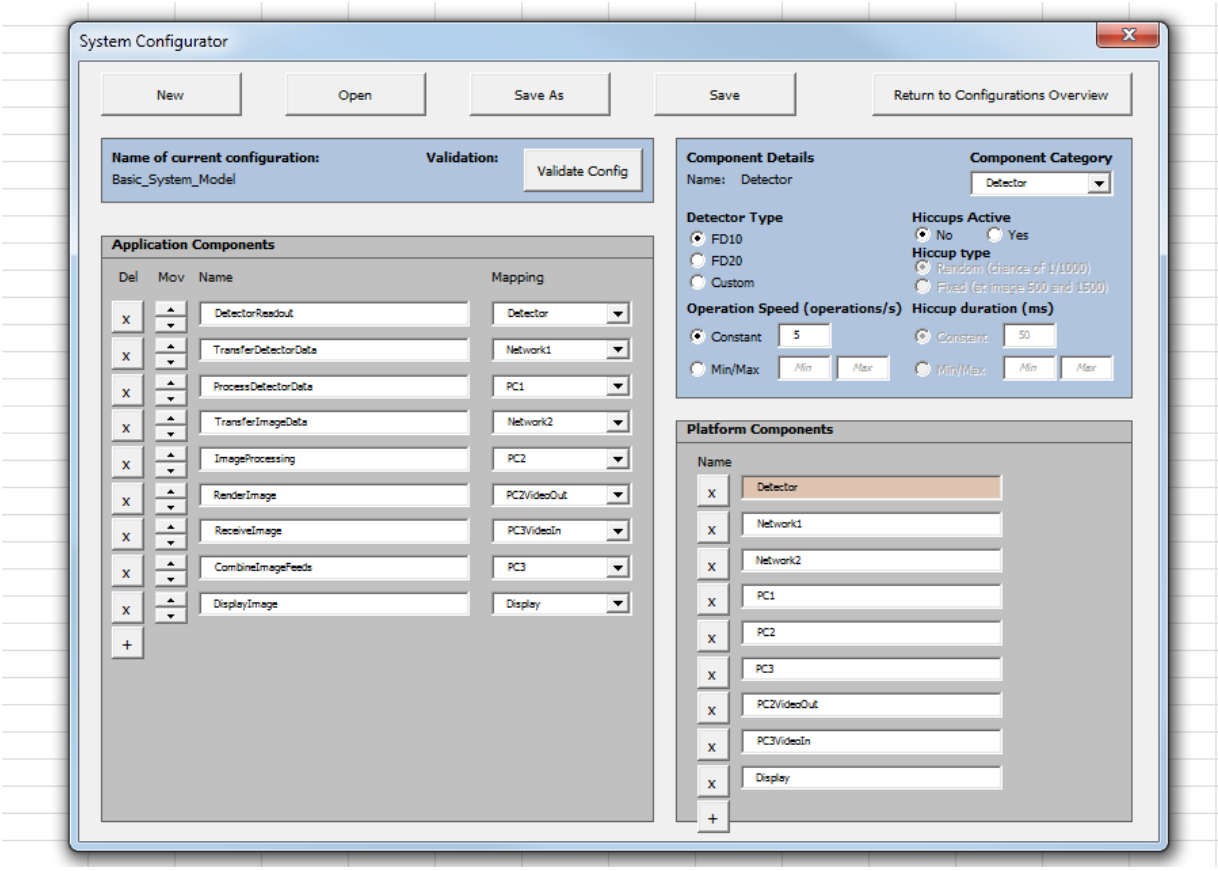


Figure B.2 – System configuration screen

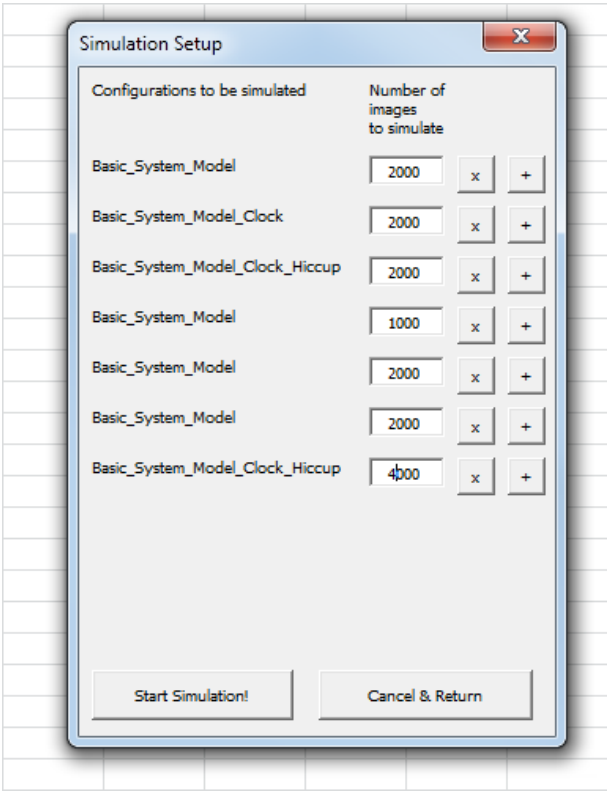


Figure B.3 – Simulation set-up screen

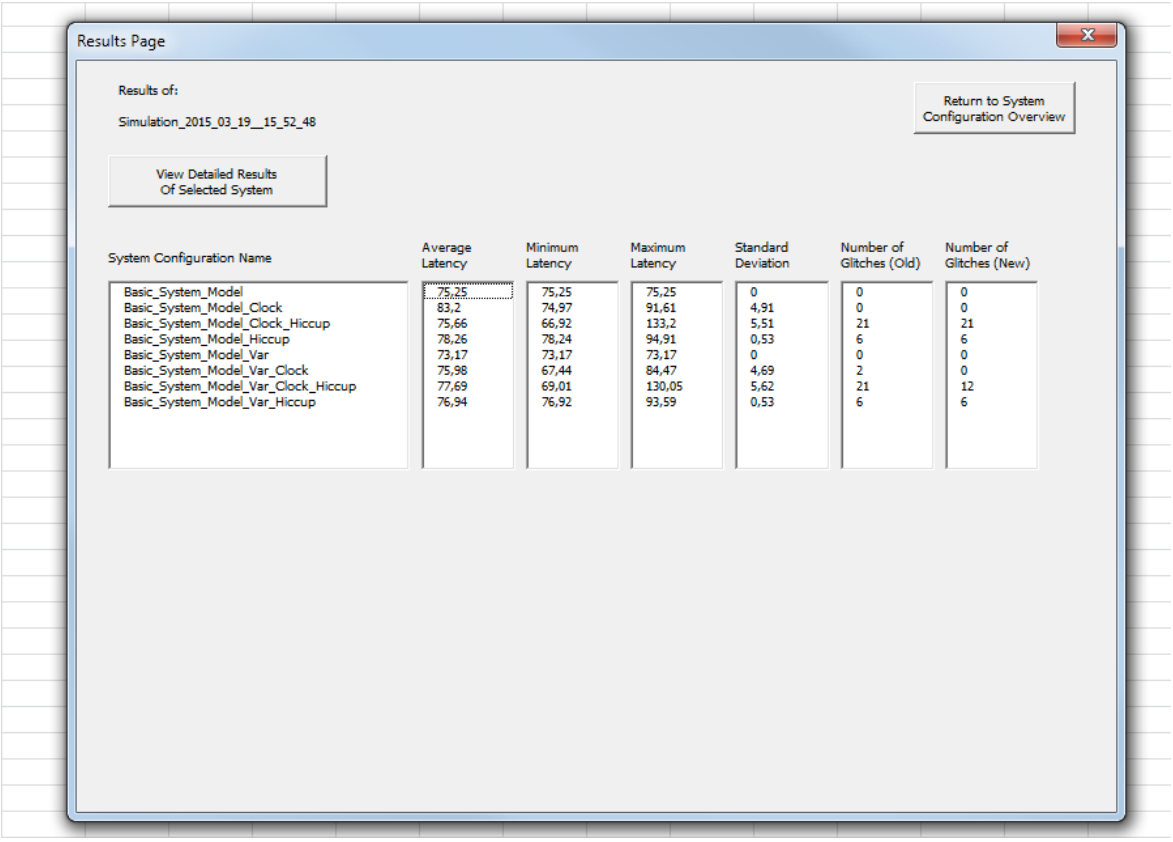


Figure B.4 – Results overview

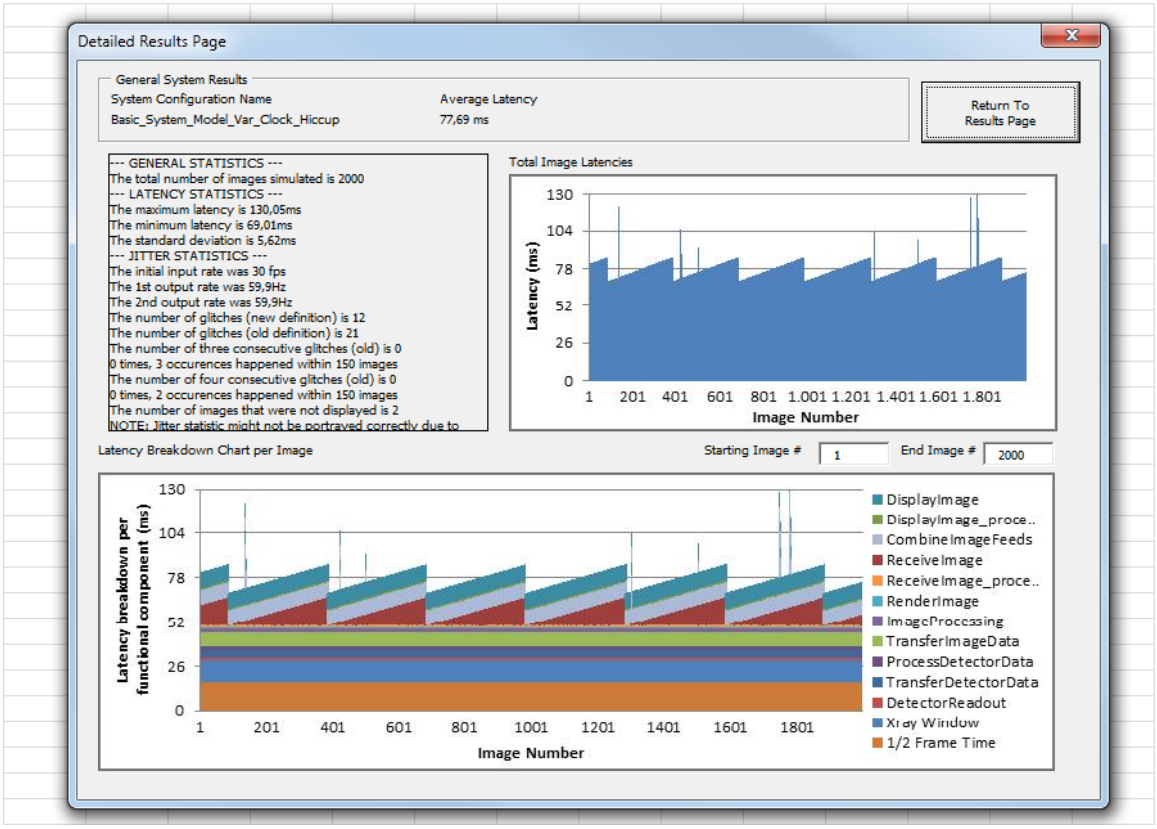


Figure B.5 – Detailed results screen

B.2 Latency & Jitter Simulation Results

The simulation results shown in this appendix support the research presented in section 5.2 and are based on the system model shown in Figure 5.6.

B.2.1 Exploration of sources of variation

To explore and understand the influence of various sources of variation on jitter and ultimately glitches, various variations on the system model presented in Figure 5.6 were explored. It was chosen to use three sources of variations in different combinations. These sources of variations are:

- **Clock change**: the value for the last two display refresh rates were changed from 60Hz to 59,9Hz
- **Hiccup**: a holdup of 50ms has a random chance of 1/1000 to occur in a process executed on PC-2 or PC-3
- **Variation**: a variation in the operation speed of PC-2 and PC-3 was introduced of 10% with a uniform distribution. This means that PC-2 will have an operation speed between 27 and 33 instead of 30 and PC-3 an operation speed between 9 and 11 instead of 10.

In Table B.1, an overview can be seen of the simulation runs done in this experiment, while Figure B.6 to Figure B.13 show the graphs corresponding to these runs.

Table B.1 – Single run simulation results for various system configurations

#	System Configuration Name	Average Latency (ms)	Minimum Latency (ms)	Maximum Latency (ms)	Standard Deviation (ms)	Number of Glitches
1	Baseline	75,25	75,25	75,25	0	0
2	Clock	83,2	74,97	91,61	4,91	0
3	Clock_Hiccup	75,66	66,92	133,2	5,51	21
4	Hiccup	78,26	78,24	94,91	0,53	15
5	Variation	73,17	73,17	73,17	0	0
6	Variation_Clock	75,98	67,44	84,47	4,69	2
7	Variation_Hiccup	76,94	76,92	93,59	0,53	9
8	Variation_Clock_Hiccup	77,69	69,01	130,05	5,62	21

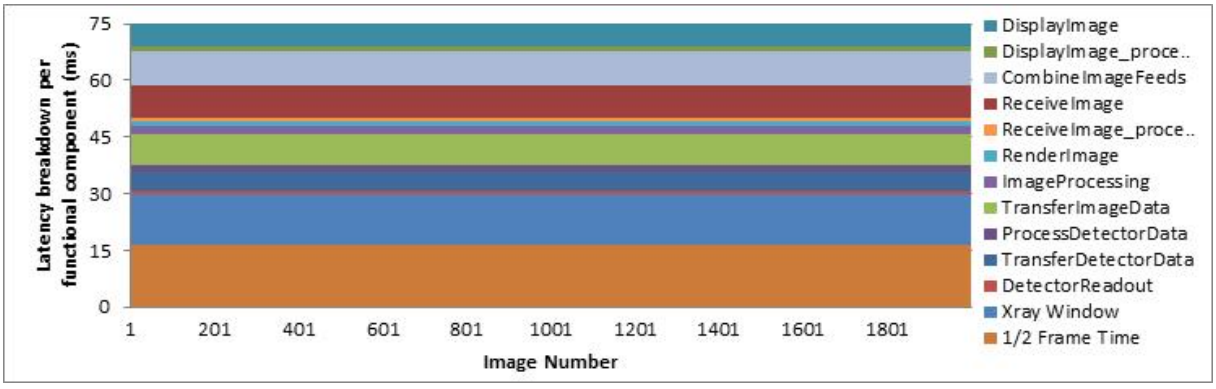


Figure B.6 – Simulation run 1: Baseline Model

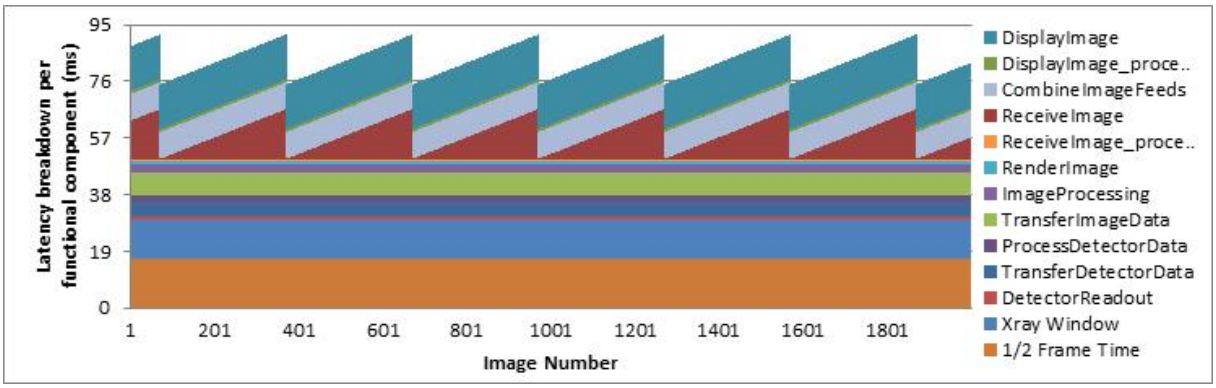


Figure B.7 – Simulation run 2: Last 2 clocks changed from 60 to 59,9Hz

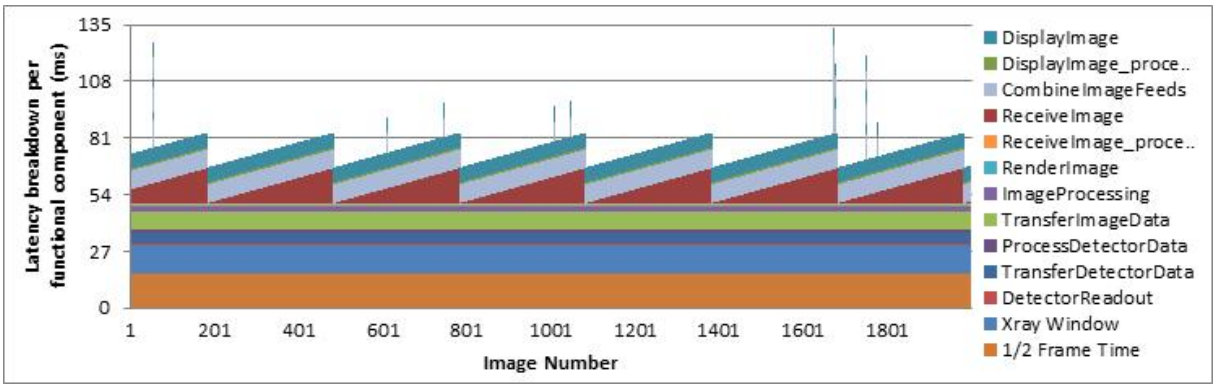


Figure B.8 – Simulation run 3: Last 2 clocks changed from 60 to 59,9Hz & Hiccups enabled

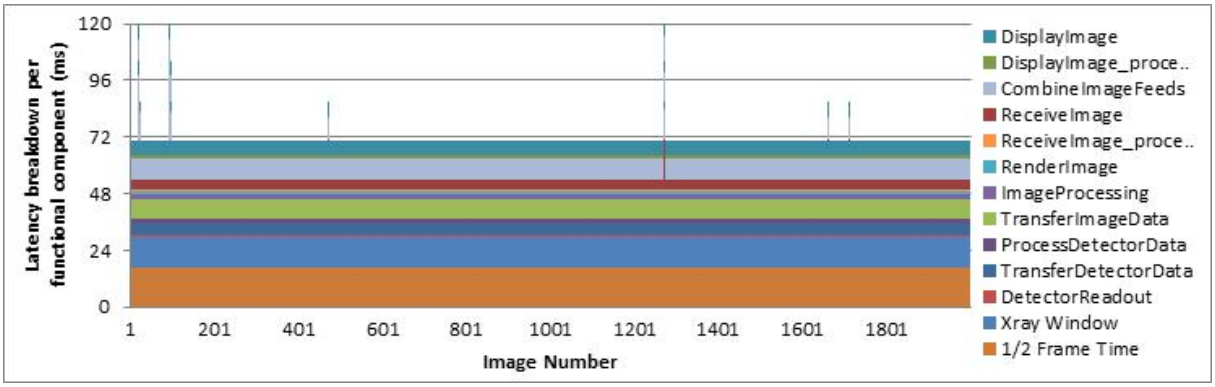


Figure B.9 – Simulation run 4: Hiccups enabled

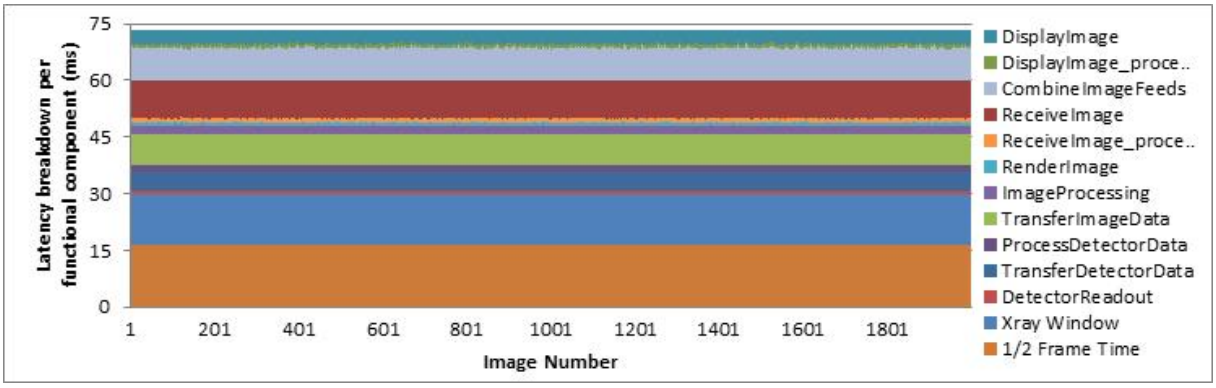


Figure B.10 – Simulation run 5: Uniform variation of 10% in PC-2 and PC-3

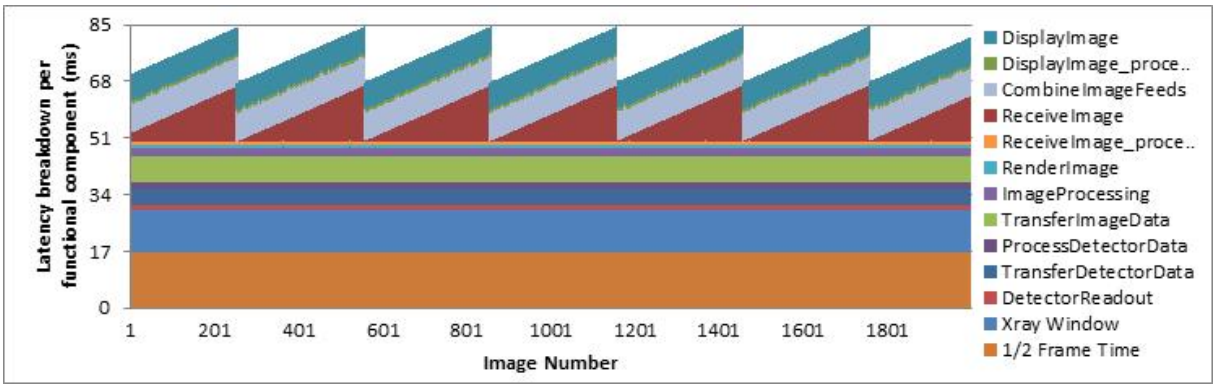


Figure B.11 – Simulation run 6: Uniform variation of 10% in PC-2 and PC-3 & last 2 clocks changed from 60 to 59,9Hz

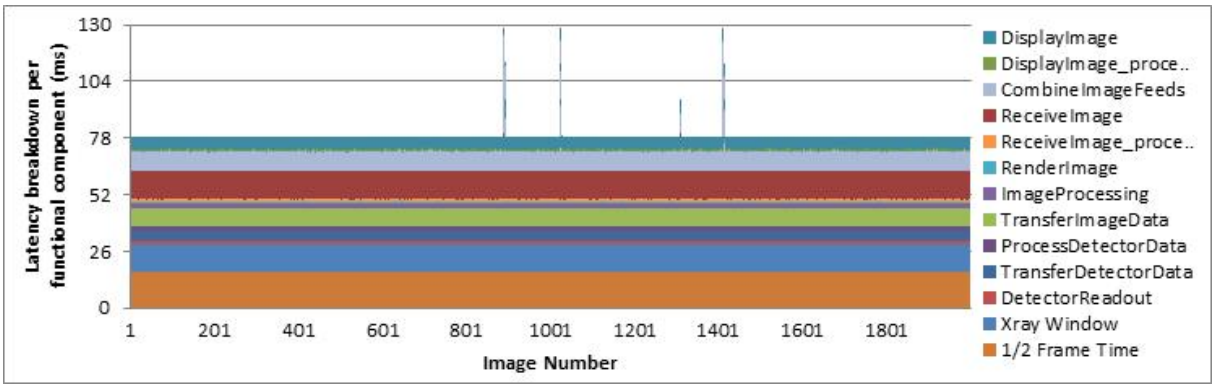


Figure B.12 – Simulation run 7: Hiccups enabled & uniform variation of 10% in PC-2 and PC-3

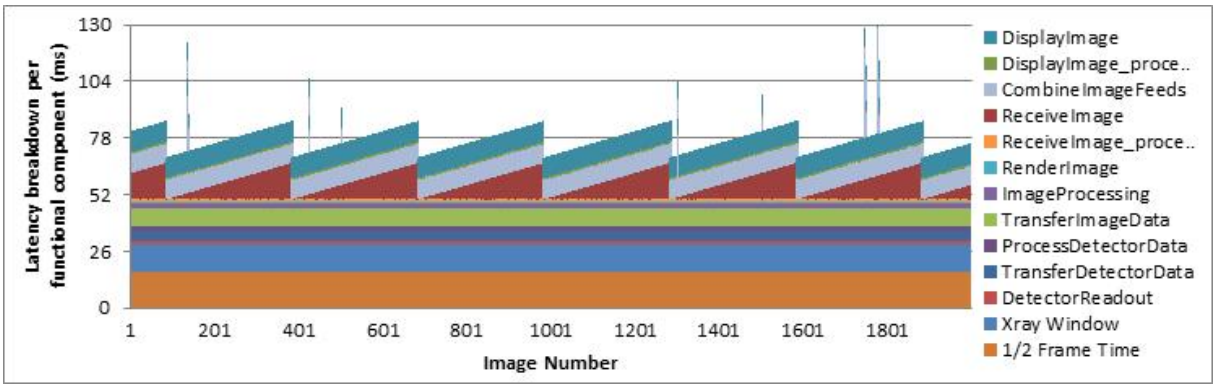


Figure B.13 – Simulation run 8: Hiccups enabled & uniform variation of 10% in PC-2 and PC-3 & last 2 clocks changed from 60 to 59,9Hz

The above graphs show various behaviors. For example, the results of these simulation runs (each configuration was run 10 times) showed that hiccups are an independent source of jitter. Also, it can be seen that the change in clock speeds results in a saw tooth like pattern. This saw tooth pattern was further analyzed and by varying the clock speeds with other values as well. As the periods of subsequent clocks are not perfect divisors, their periods are not aligned. Normally, in the 30 FPS and 60 Hz case, each image is shown two times on the display, as every two periods of the display, exactly one new image arrives, always with the same waiting time until the next refresh cycle. This is also shown in the left side of Figure B.14.

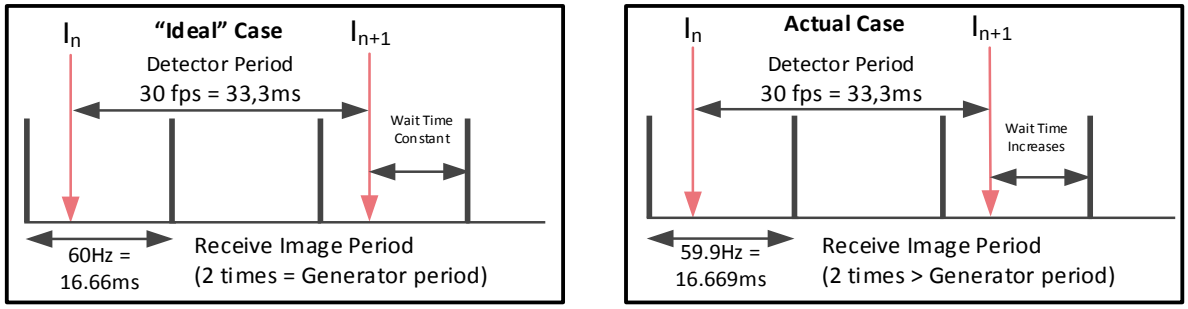


Figure B.14 – Saw tooth explanation. Left is the ideal case with no saw tooth, right is the actual case, with saw tooth behavior.

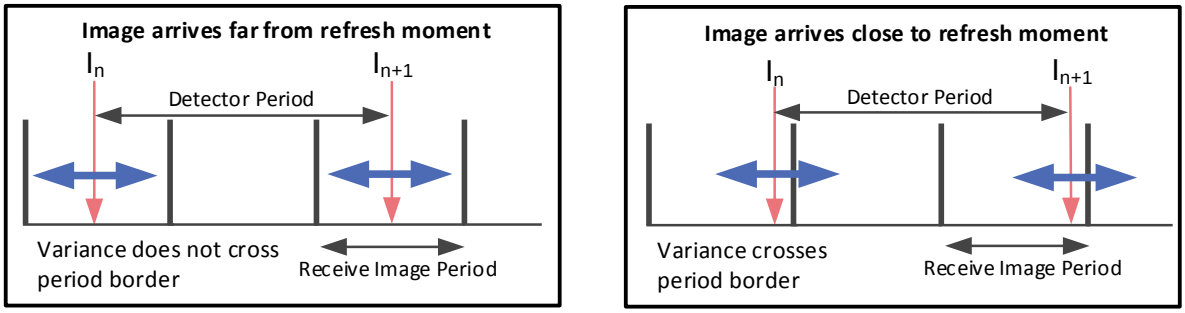


Figure B.15 – Glitch explanation. In the left situation no glitches occur, in the right situation glitches do occur

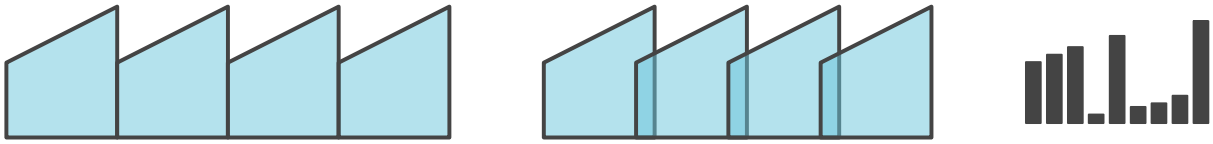


Figure B.16 – Illustration of relation between "saw teeth" and glitches.

However, if the periods are unaligned, the wait time will either increase or decrease. At a certain point, the image will either skip a refresh cycle, or end up in an earlier one. This is shown in the right side of Figure B.14. Nonetheless, the saw tooth behavior does not cause glitches independently, but only if there is also variation in the system. The same holds true for small variations, as they are buffered by the constant wait time. Once the adjusted clocks and variation are present in the system together, glitches start to occur. The explanation to this is illustrated in Figure B.15, where it can be seen that there is no single moment where an image ends up in the "wrong" refresh cycle, but this happens multiple times in the period that the image arrival time is around the refresh moment.

Figure B.16 illustrates this concept. On the left side, separate "saw teeth" do not overlap, but in the glitch case they do overlap, as can be seen in the middle. This results in the behavior on the right side where differences between subsequent images arise.

B.2.2 Exploration of impact of variation size

Another issue that was explored is the size of the variation. In the system model that was used for this simulation no hiccups were introduced and the clocks of the display components were both set to 59,9Hz. The variation percentage was changed over the runs. For each system model, 10 runs were executed. Table B.2 shows the number of glitches for

each run and it can be seen that there are a number of surprising cases with an abnormal amount of glitches. For example, run number 7 for the 5% variation case has 108 glitches. These outliers are explained in the main body of this thesis, which is section 5.2.1.

Furthermore, Table B.3 list the average amount of glitches per variation percentage and Figure B.17 maps the average number of glitches without outlier in a graph. It can be seen that there is a linear relation between the variance and the number of glitches.

Table B.2 – Number of glitches per simulation run for various variation percentages

		Number of glitches for each run number									
#	**System Config. Name**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0% Variation	0	0	0	0	0	0	0	0	0	0
2	1% Variation	0	0	0	0	0	0	0	0	0	0
3	5% Variation	0	4	2	3	0	1	108	6	0	1
4	10% Variation	176	7	5	5	3	3	6	157	22	3
5	15% Variation	169	7	3	10	57	5	2	4	6	5
6	20% Variation	6	5	11	8	6	8	196	9	223	0
7	25% Variation	54	10	6	14	8	10	10	11	218	10
8	50% Variation	20	31	27	206	17	33	183	14	82	122

Table B.3 – Average number of glitches per variance percentage

Variance	**Average number of glitches in 10 runs of 2000 images**	
	With outliers	**Without outliers**
0%	0	0
1%	0	0
5%	12,5	1,9
10%	38,7	4,6
15%	26,8	5,3
25%	47,2	6,6
25%	35,1	9,9
50%	73,5	23,7

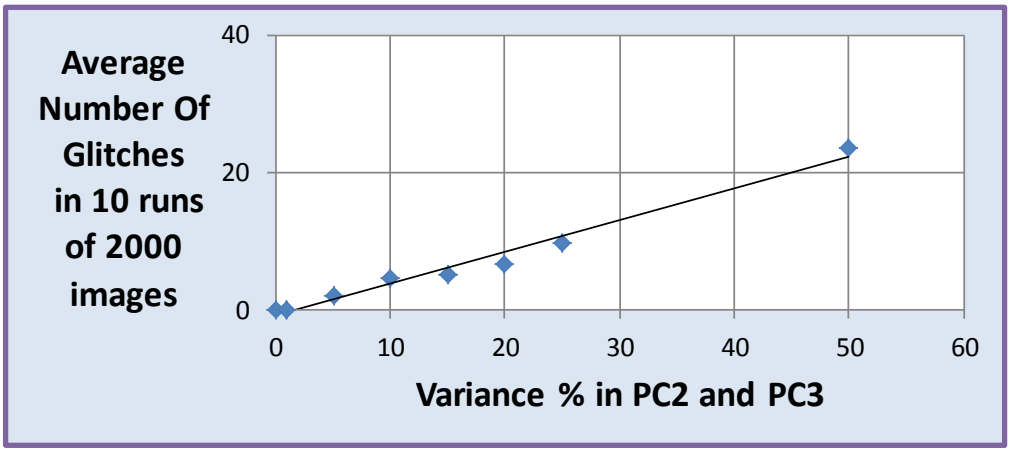


Figure B.17 – Average number of glitches without outliers versus variance percentage

Appendix C Power Distribution Case Study Details

This appendix provides further information for the Philips iXR Power Management case study described in section 5.3. Three types of views were detailed for each system state. The functional view, the physical view and the context view.

C.1 Functional Views

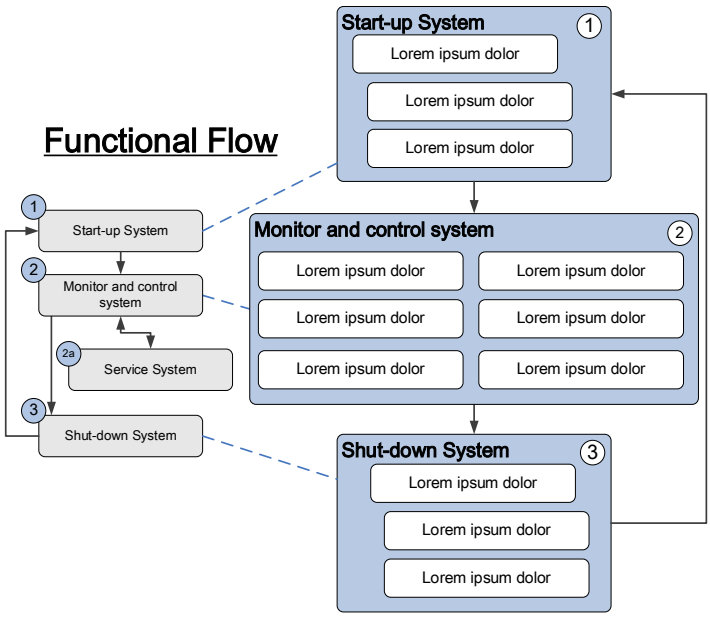


Figure C.1 – Functional View for the three following system states: "Mains Switch Off", "System Standby On Battery" and "System Standby On Mains"

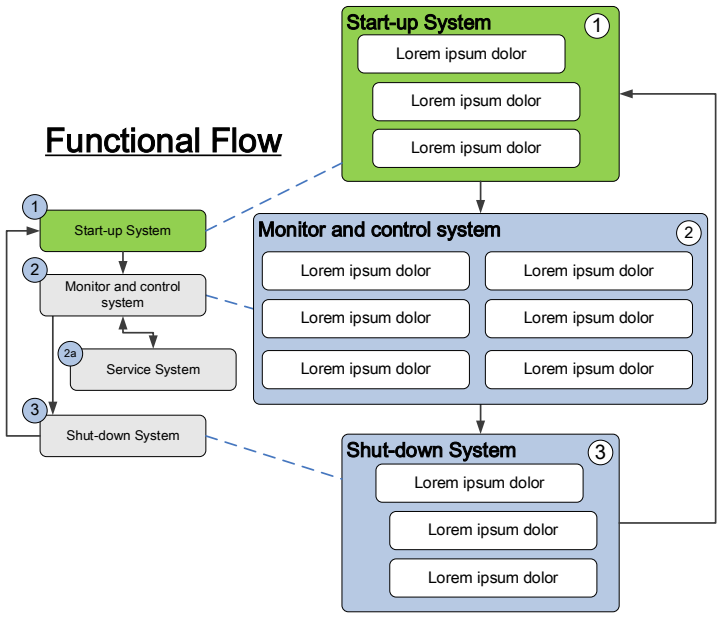


Figure C.2 – Functional View for the two following system states: "Starting System On Mains" and "Starting System On Battery"

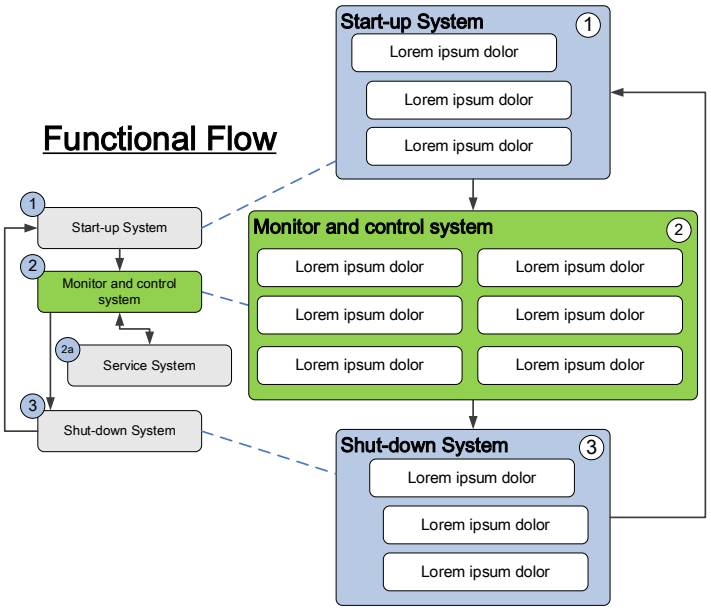


Figure C.3 – Functional View for the four following system states: "System On On Mains", "System On On Battery", "Segment A On On Mains" and "Segment A On On Battery"

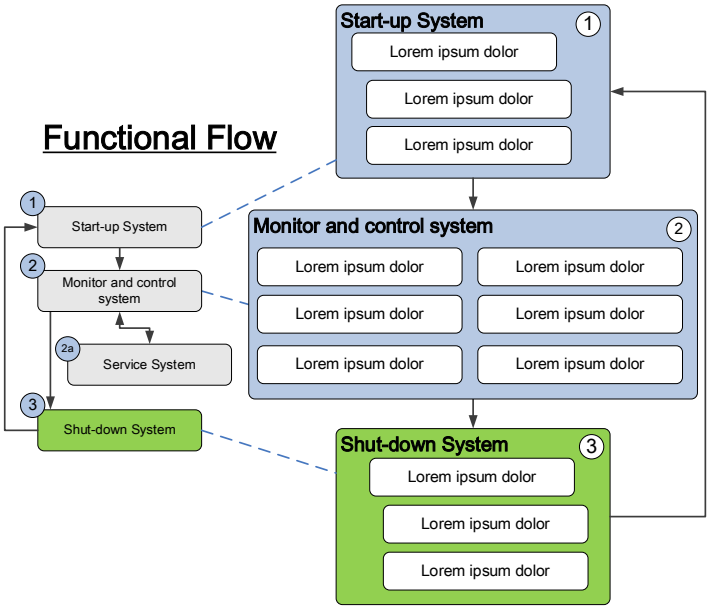


Figure C.4 – Functional View for the three following system states: "Shutting Down On Mains", "Shutting Down On Battery" and "Shutting Down Battery Low"

C.2 Physical Views

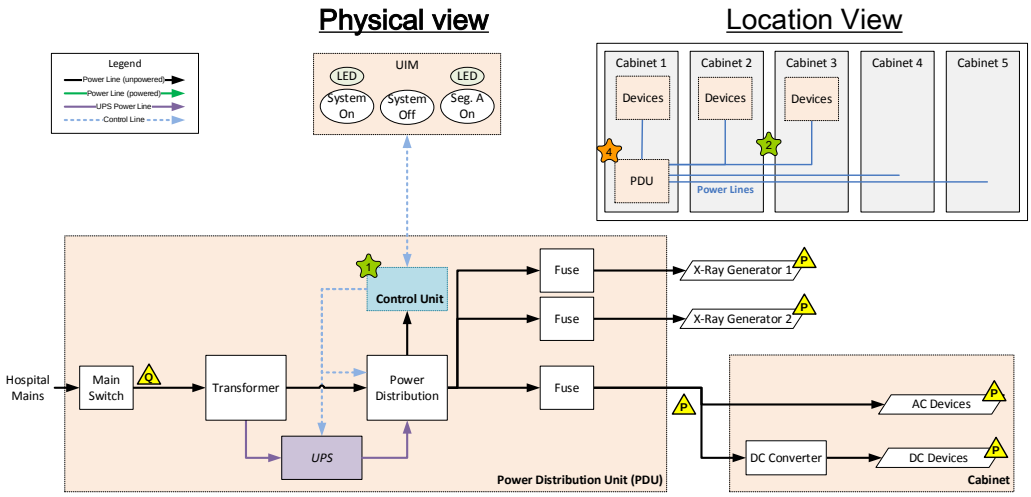


Figure C.5 – Physical View for state: "Mains Switch Off"

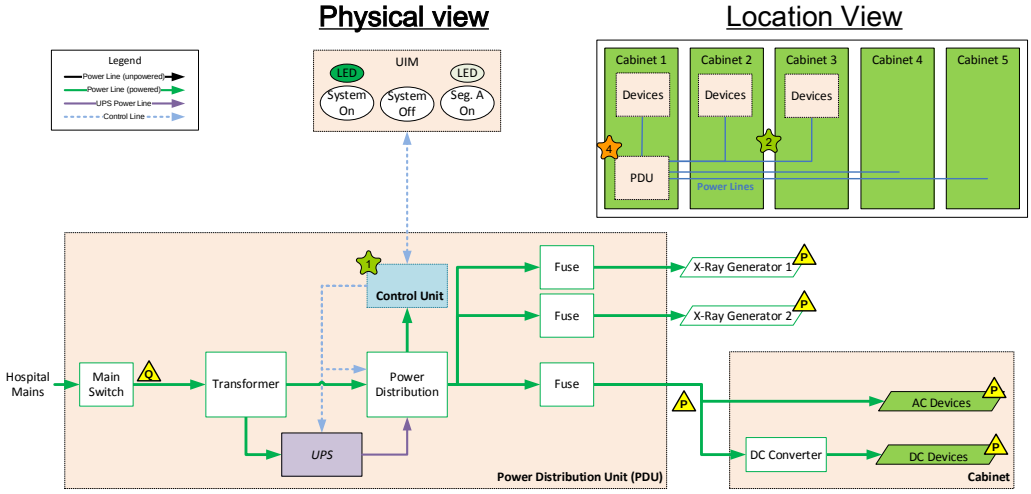


Figure C.6 – Physical view for the three following states: "Starting System On Mains", "System On On Mains" and "Shutting Down On Mains"

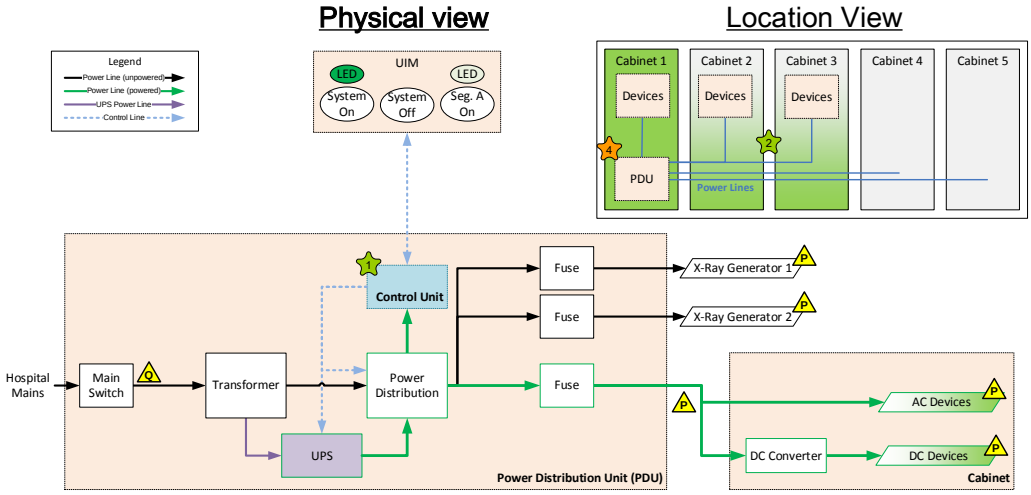


Figure C.7 – Physical view for the four following states: "Starting System On Battery", "System On On Battery", "Shutting Down On Battery" and "Shutting Down Battery Low"

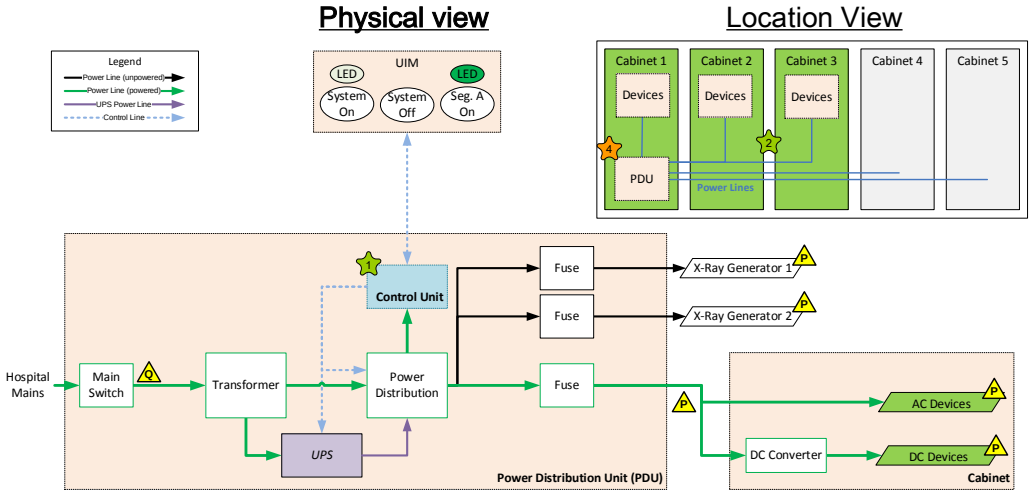


Figure C.8 – Physical View for state: "Segment A On On Mains"

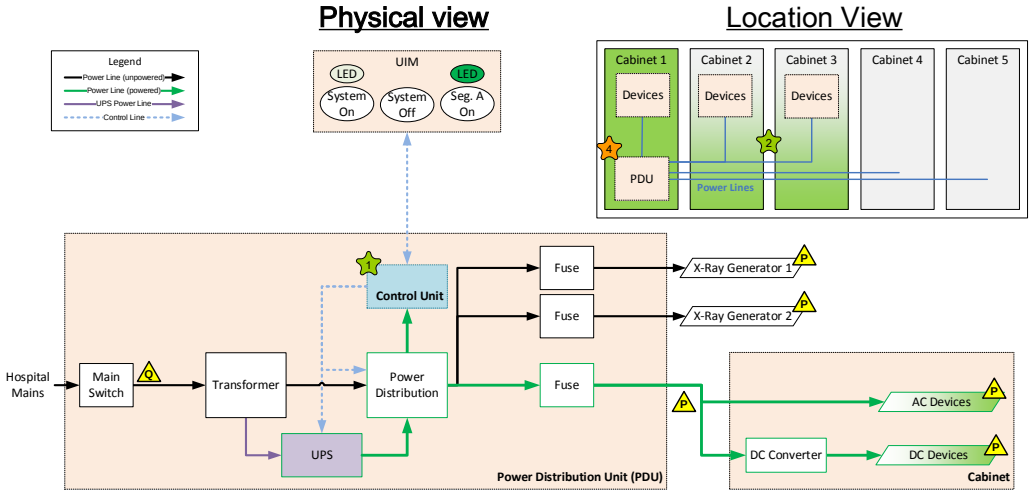


Figure C.9 – Physical View for state: "Video On On Battery"

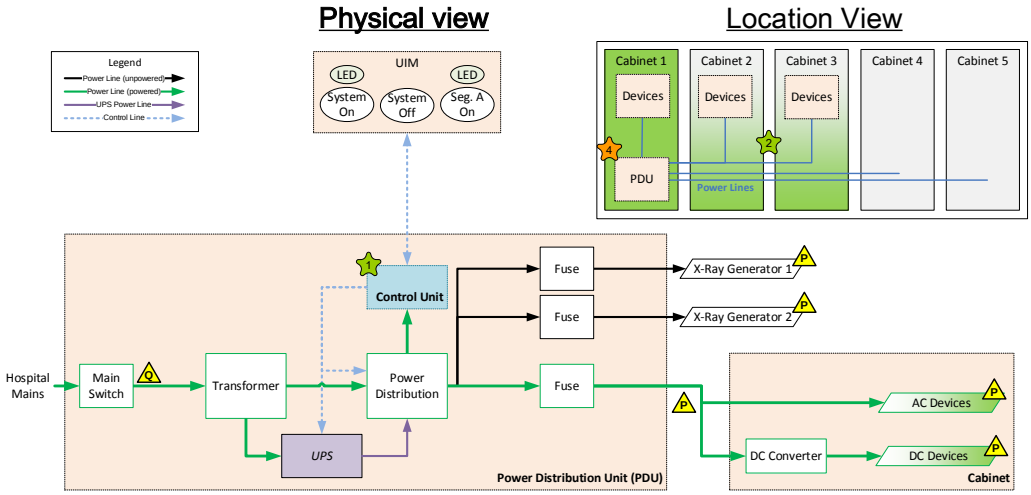


Figure C.10 – Physical View for state: "System Standby On Mains"

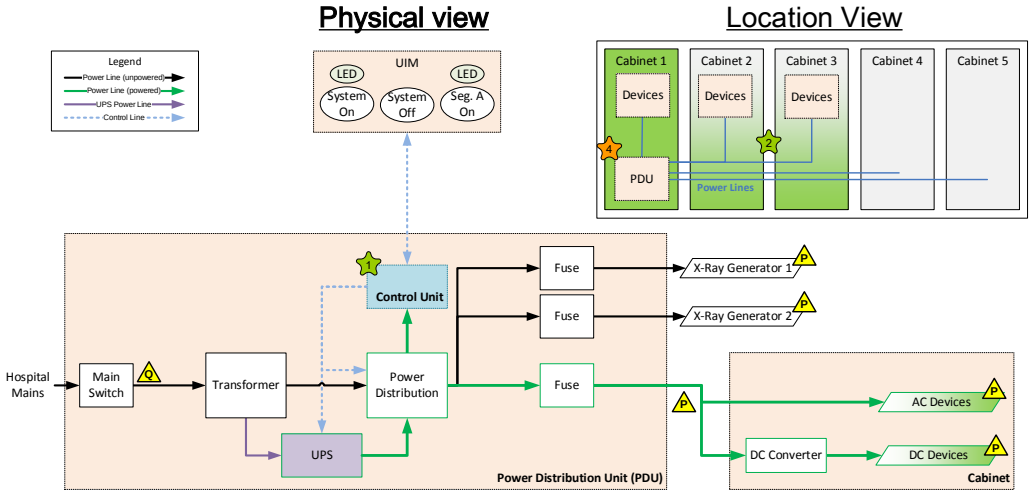


Figure C.11 – Physical View for state: "System Standby On Battery"

C.3 Context Views

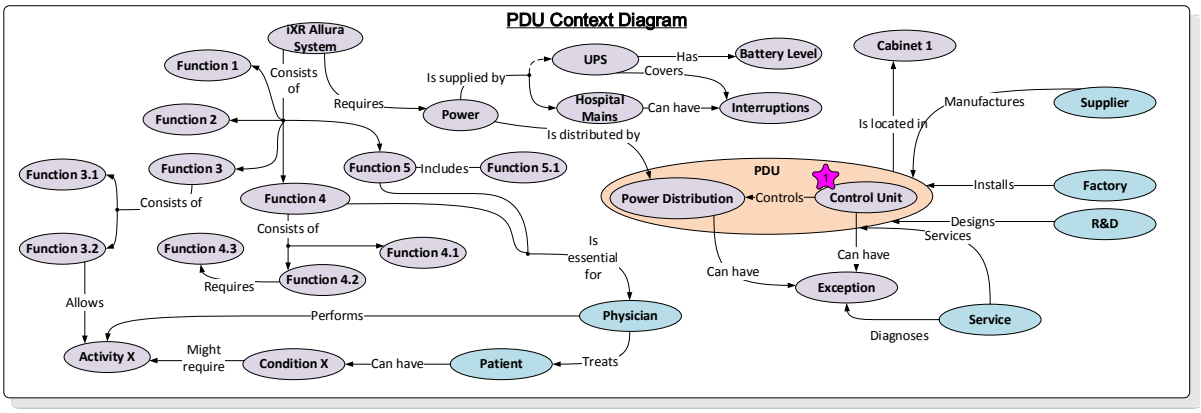


Figure C.12 – Context View for state: "Mains Switch Off"

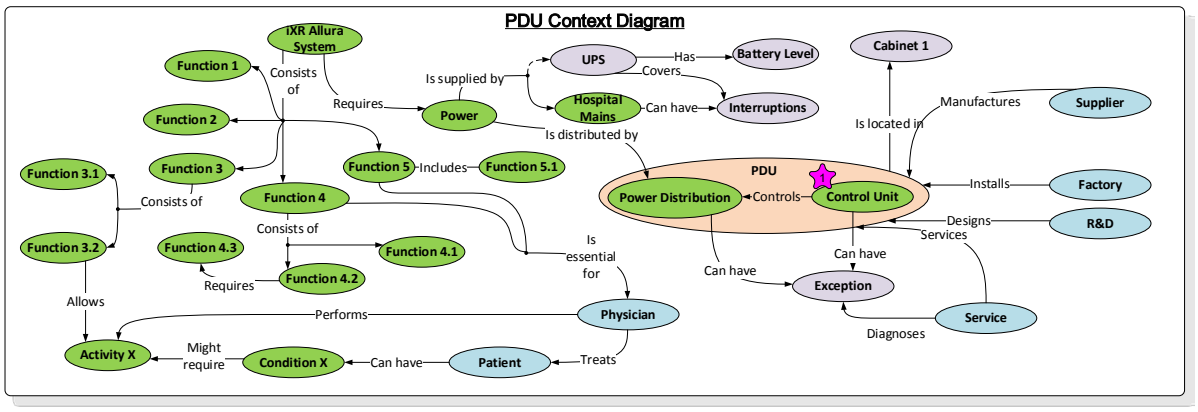


Figure C.13 – Context View for state: "System On On Mains"

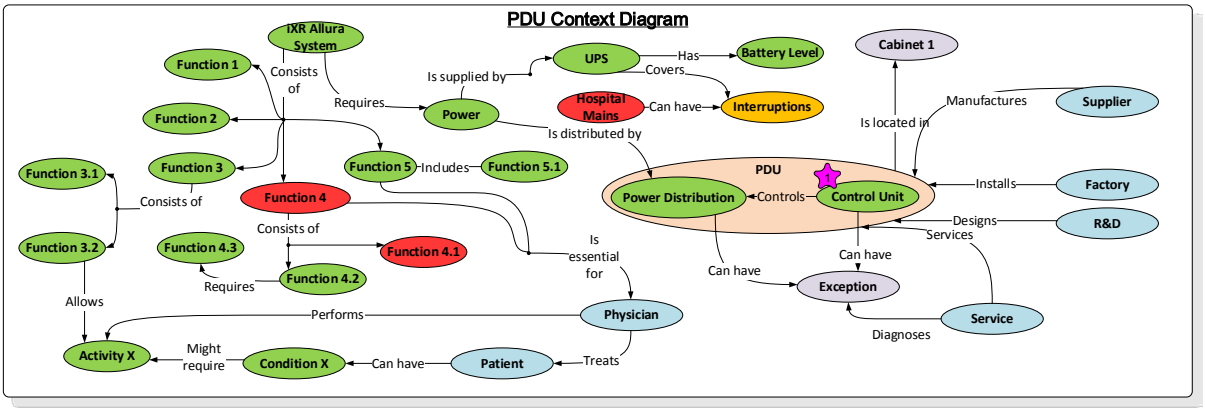


Figure C.14 – Context View for state: "System On On Battery"

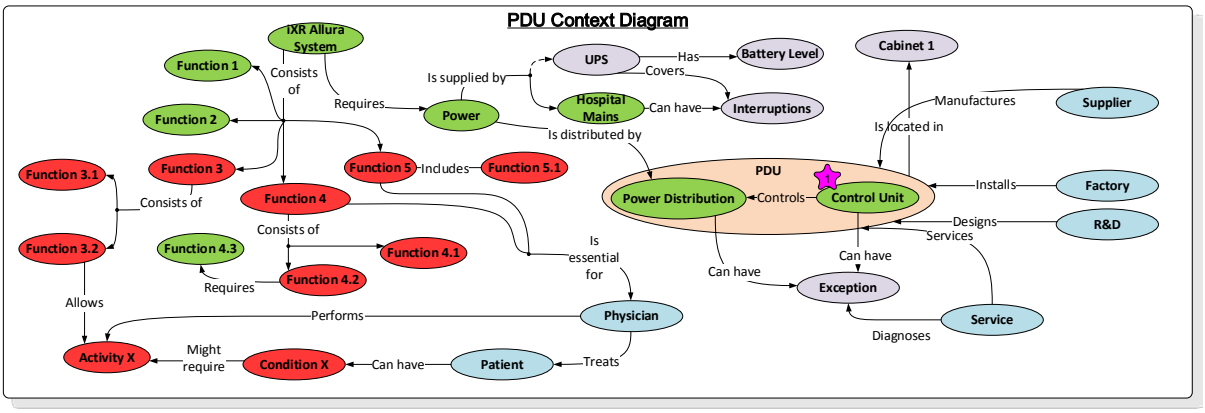


Figure C.15 – Context View for the following three states: "Starting System On Mains", "System Standby On Mains" and "Shutting Down On Mains".

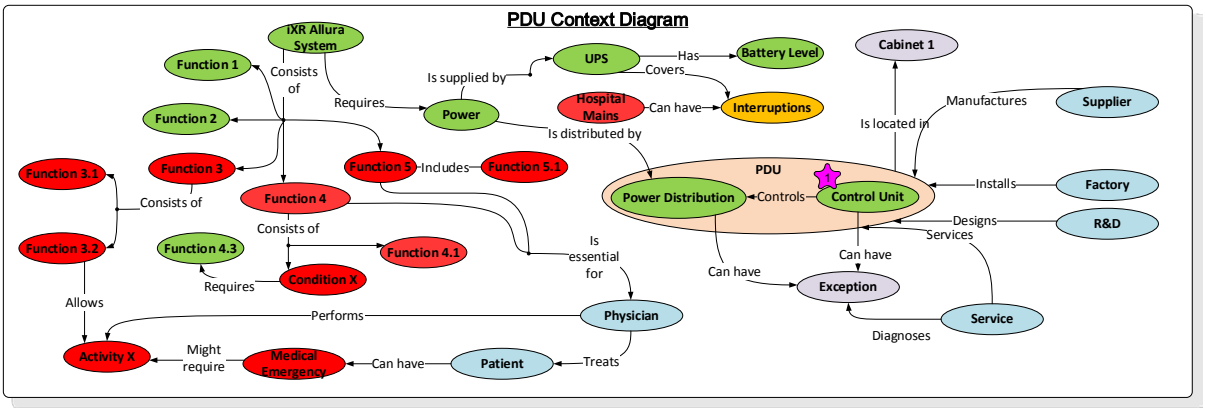


Figure C.16 – Context View for the following four states: "Starting System On Battery", "System Standby On Battery", "Shutting Down On Battery" and "Shutting Down Battery Low".

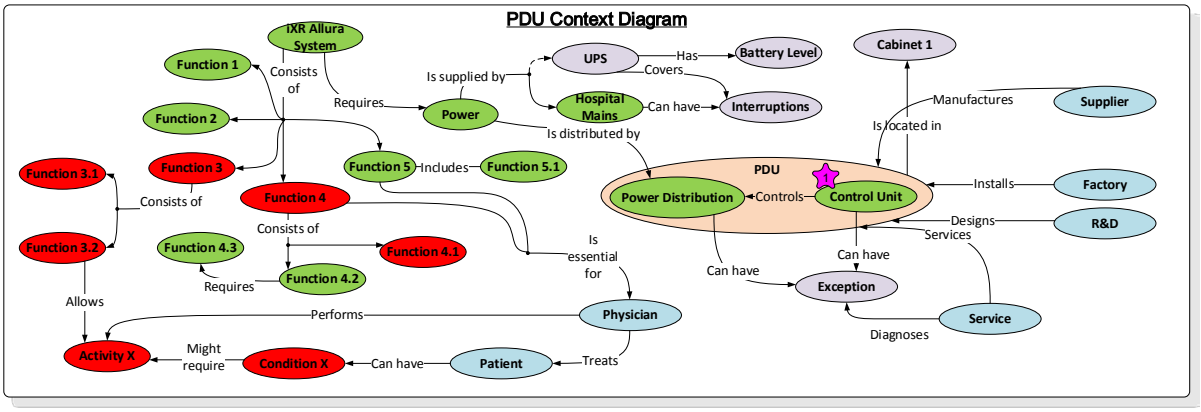


Figure C.17 – Context View for state: "Segment A On On Mains"

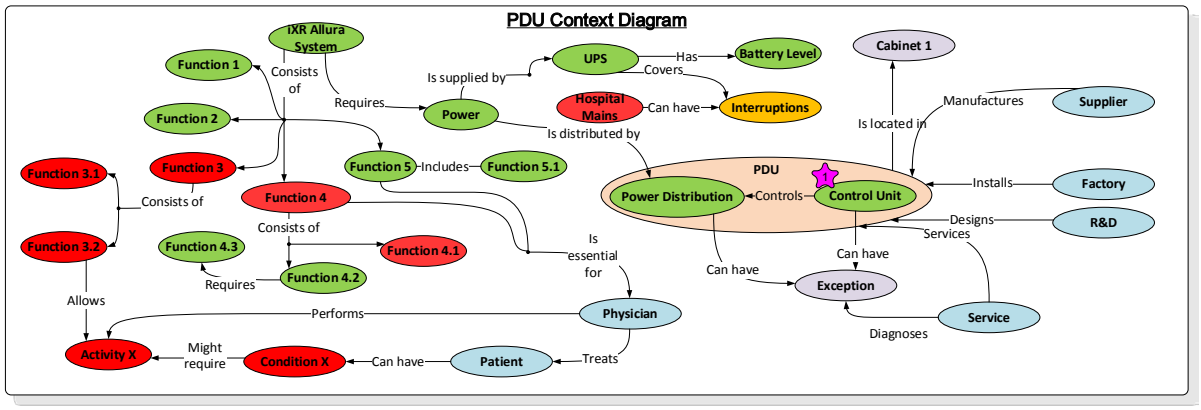


Figure C.18 – Context View for state: "Video On On Battery"

Appendix D Resources for Application of the COMBOS Method

This appendix provides further resources for application of the COMBOS Method. It summarizes the guidelines mentioned throughout Chapter 6. These guidelines consolidate the experiences and knowledge used to develop the method. The guidelines outline how to enable and support the communication of system behavior in conceptual system design. It also provides a detailed stepwise description of how to execute COMBOS, including references to methods used.

Table D.1 – Guidelines to enable and support the communication of system behavior

#	**Guideline**
1	Exploring system behavior (using this method) is mainly useful if the conceptual design process does not allow a "quick and dirty" way to test the new system via a prototype
2	While a simulation engineer might be required to create an interactive system overview, the system architect is responsible for deciding which viewpoints are used to construct views in the interactive system overview
3	To represent and communicate the operational view, it must be coupled to other views, such as the physical, functional, quantification or context view
4	Visualizing the operational view simultaneously in different views allows reasoning on system behavior from different perspectives, which in turn enables multidisciplinary design discussions, and ultimately, new knowledge creation
5	Ensure continuous communication throughout the process with affected stakeholders, especially the decision makers, focusing on construction and validation of the created views in each step
6	The structural views for the concerns that are elaborated with behavioral views must be established. It is also highly recommended to create structural views for all concerns, as this ensures that the full system architecture is considered
7	Once a need for behavioral analysis is identified, translate this need in well-defined goals for the analysis process. These goals should act as a reference during creation of the interactive system overviews
8	Keep track of the questions that stakeholders have with respect to the system behavior. If the number of questions stabilizes, either due to the fact that they are answered or cannot be answered, stop the analysis or transition to the next step
9	When considering the behavior of a system across states, both the behavior of a system in that state, as well as the possible transitions to other states should be subject of interest. Simulation models explicitly allow exploration of these transitions
10	To communicate an interactive system overview, preserve the size of a paper based A3 Architecture Overview (A3 size). Therefore, aim to use (portable) devices that have a diagonal screen size of 18-22 inch and support resolutions of 1920x1080 or higher
11	When establishing simulation models in conceptual system design, aim to model only generic behavior and avoid going into too much detail as this distracts from exploration of the problem domain

Table D.2 – Stepwise COMBOS Process Overview

Task	Sub Steps	Approach	Output
Step 1 - Create Views for Overall System Concern and Identify Other Concerns			
Identify Overall System Concern (Goal)	Create Context View & List Stakeholders	Systems Thinking [Boardman and Sauser, 2008]	Systemigram
	Create Operational View	Story Telling [Muller, 2004, p103]	System Usage Scenarios
	Create Functional View	Functional Analysis e.g. [Pahl *et al.*, 2007, p169]. Include utility functions [Bonnema, 2008, p59]	Functional Block Diagram or Function List
Identify Other Concerns	Identify Key Drivers	Consider Stakeholder Objectives, see also [Muller, 2004, p59]	List of Key Drivers or Key Driver Map
	Identify Subsystems	FunKey Architecting [Bonnema, 2008]	FunKey Diagram
Make Concerns Overview	Consolidate Identified Concerns	See both section 3.3.2 and Figure 3.4	Concerns Overview
Step 2 - Create Structural Views for each Key Driver and Subsystem Concern			
For each Key Driver or Subsystem Concern	Create Functional, Physical, Context, Quantification and other relevant views, as well as visual aids	Follow A3 Architecture Overview creation process [Borches, 2010, p192-193], but also include systemigrams	A3 Architecture Overview for each concern
Step 3 - Create Behavioral Views with Interactive System Overviews for selected Key Driver and Subsystem Concerns (See also Section 6.3.3)			
Continuous Stakeholder Involvement	Elicit Stakeholder Feedback	Use (Interactive) System Overview to Elicit Feedback about System Behavior	List of Stakeholder Questions, Visual Aids
Identify Need For Behavioral Analysis	A. Identify Need For (More) Behavioral Analysis	Find needs using the three triggers identified in Section 6.3.3, Step 3A	Analysis Goals
	B. Identify Behavioral Characteristics	Identify whether main influences on behavior are characterized well	List of Behavioral Characteristics
Characterize System Behavior over System States	C. Identify Relevant System States	Utilize scenarios created with storytelling or consider current implementation to identify states	List of System States, State Transition Diagram
	D. Detail Overview for Each System State	Detail views by choosing interactions (see Table 6.1)	Detailed Views for each System State
	E. Construct System Behavior Simulation	POOSL Simulation [Schuts and Hooman, 2015a]	POOSL Simulation
	F. Create User Interface with System View and Visual Aids	User Interface development based on chosen interactions in Eclipse or Microsoft Excel	Interactive System Overview
Conduct Simulation Study to Uncover Behavioral Characteristics	G. Conduct Simulation Study	Y-Chart modeling [Basten *et al.*, 2013] and simulation model in POOSL or Excel	Y-Chart based System Model, Simulation Model
	H. Extend Simulation With System View	Extend user interface based on chosen interactions in Eclipse or Microsoft Excel	Interactive System Overview

Appendix E Electric Vehicle Example

This appendix provides further information for the Electric Vehicle example described in section 7.1. Three types of views were detailed for each system state. The functional view, the physical view and the context view.

E.1 Physical Views

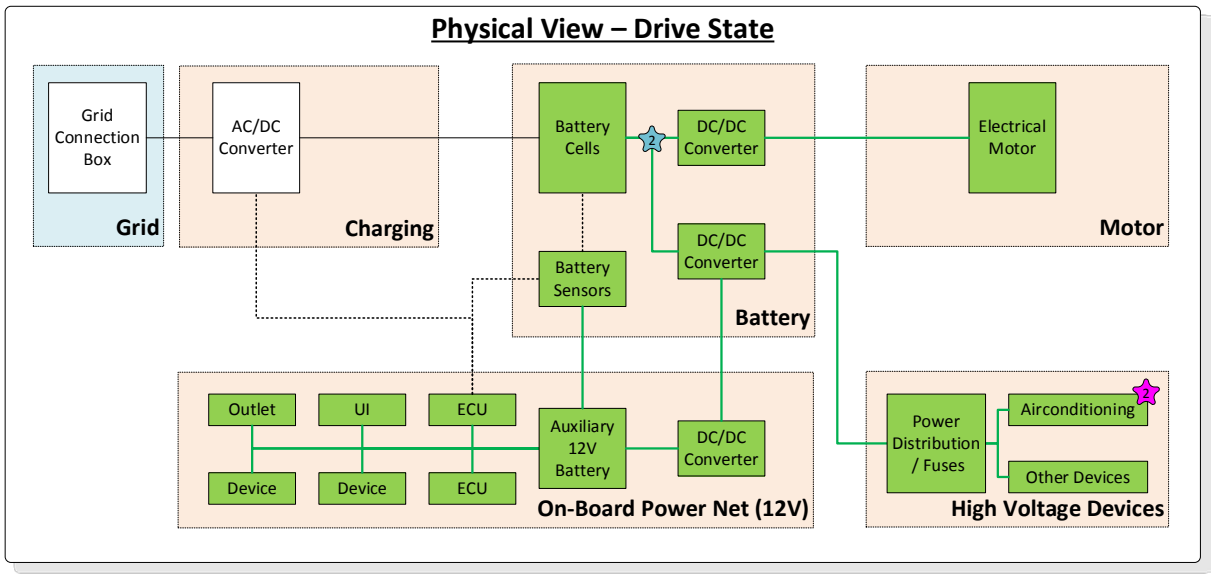


Figure E.1 – Physical View for the Drive State

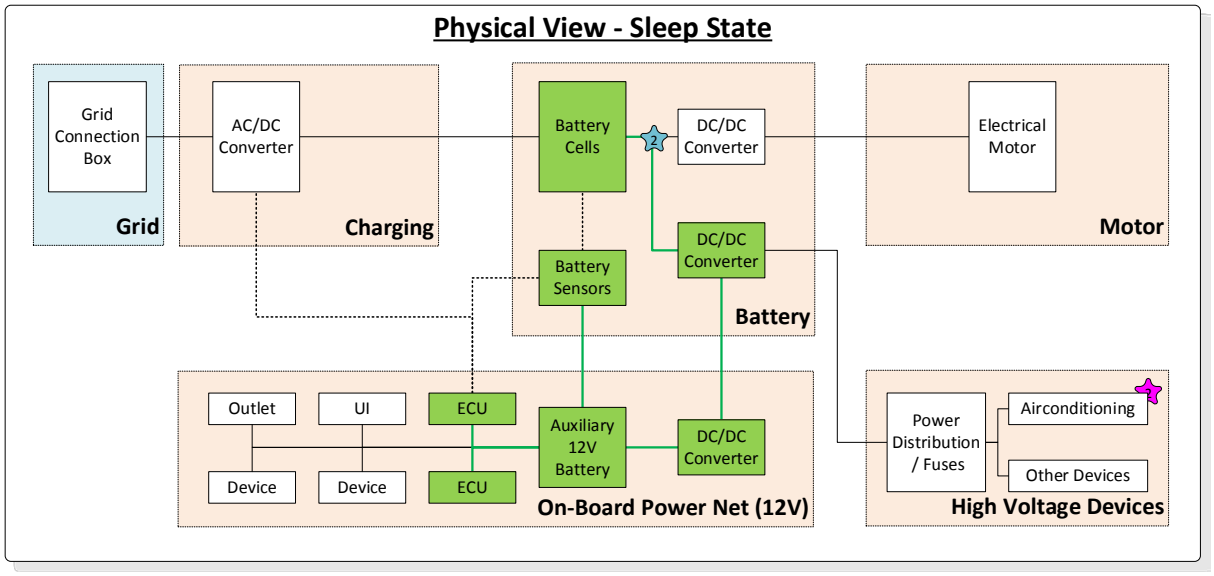


Figure E.2 – Physical View for the Sleep State

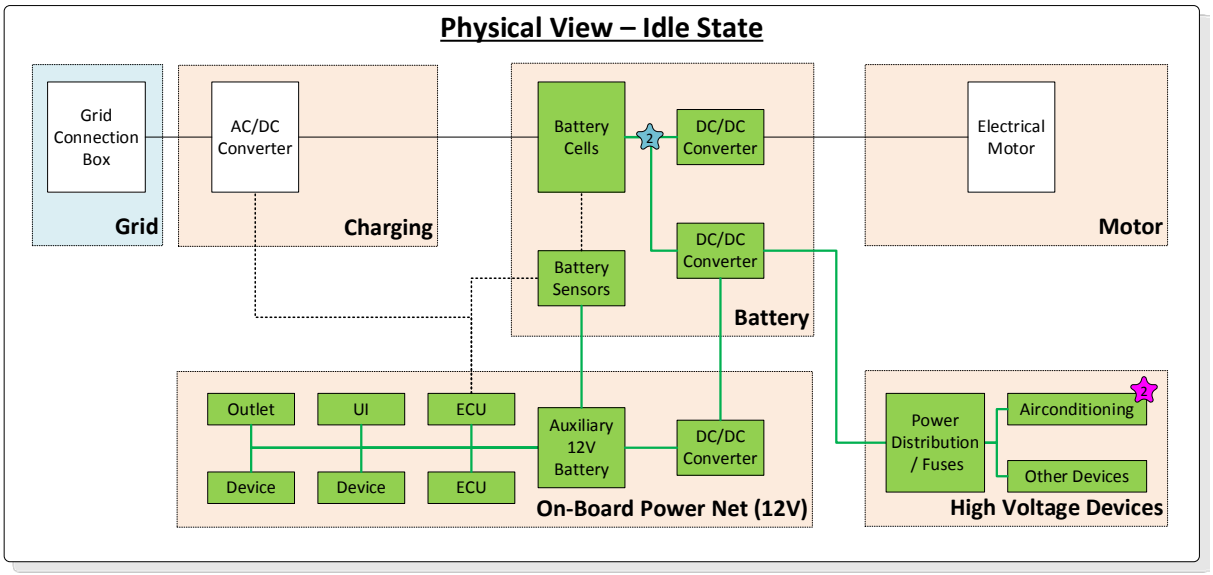


Figure E.3 – Physical View for the Idle State

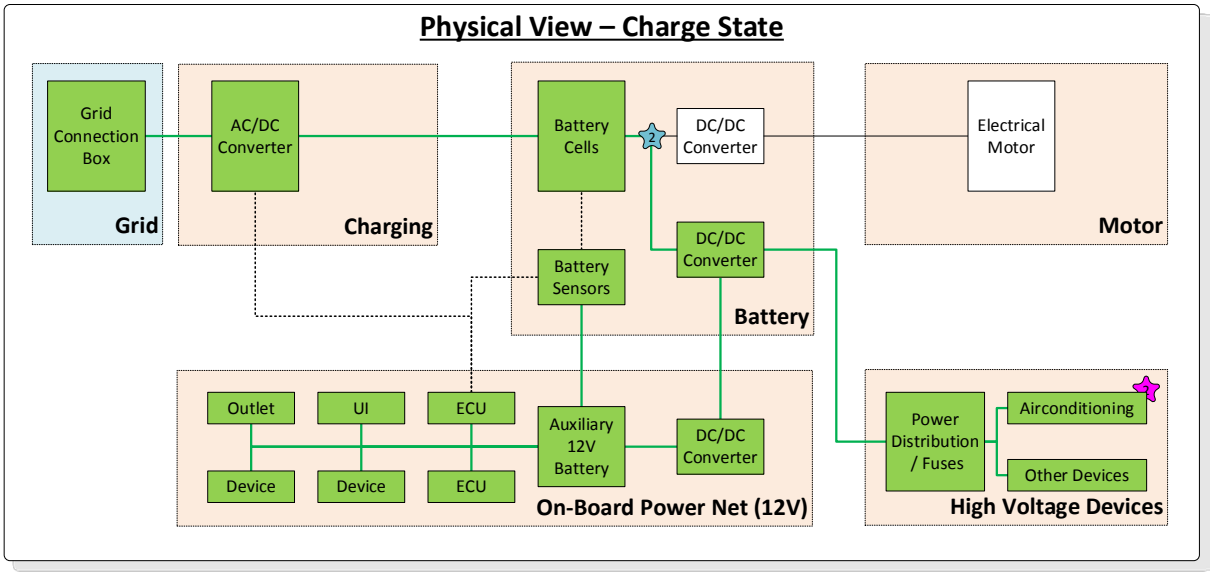


Figure E.4 – Physical View for the Charge State

E.2 Functional Views

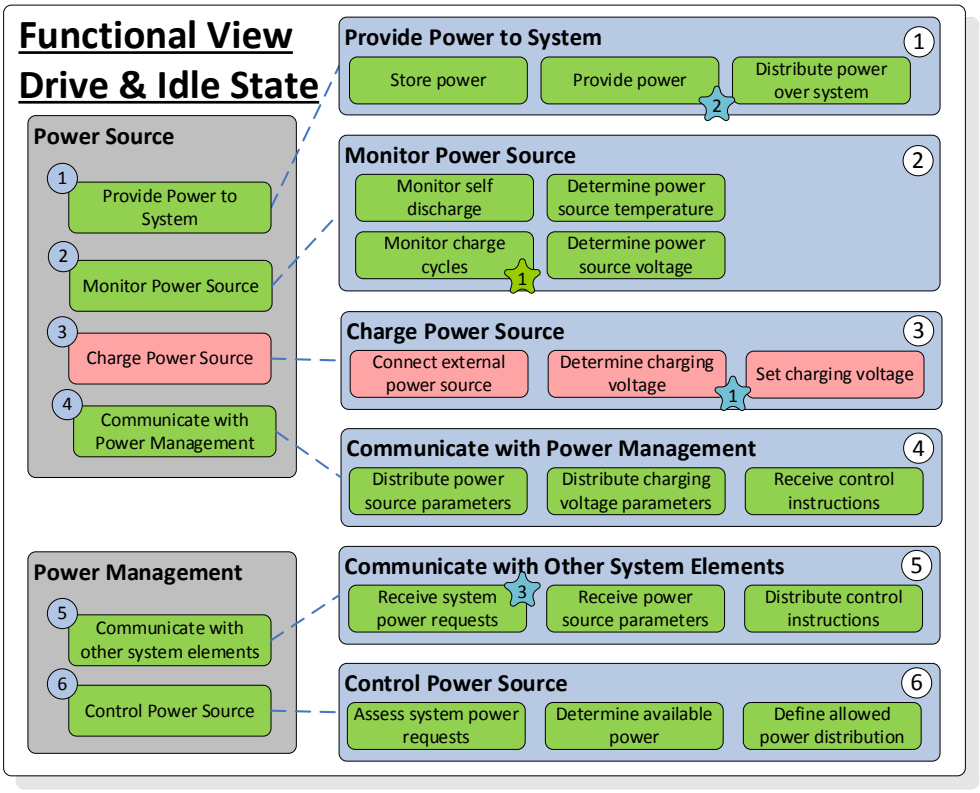


Figure E.5 – Functional View for the Drive and Idle States

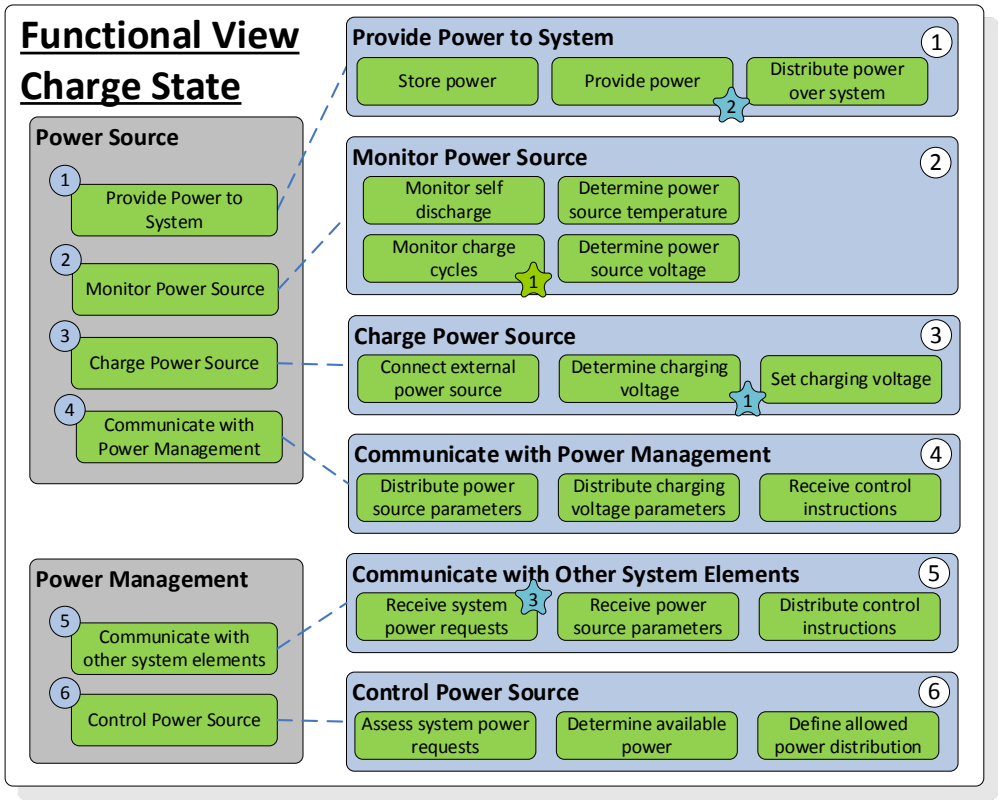


Figure E.6 – Functional View for the Charge State

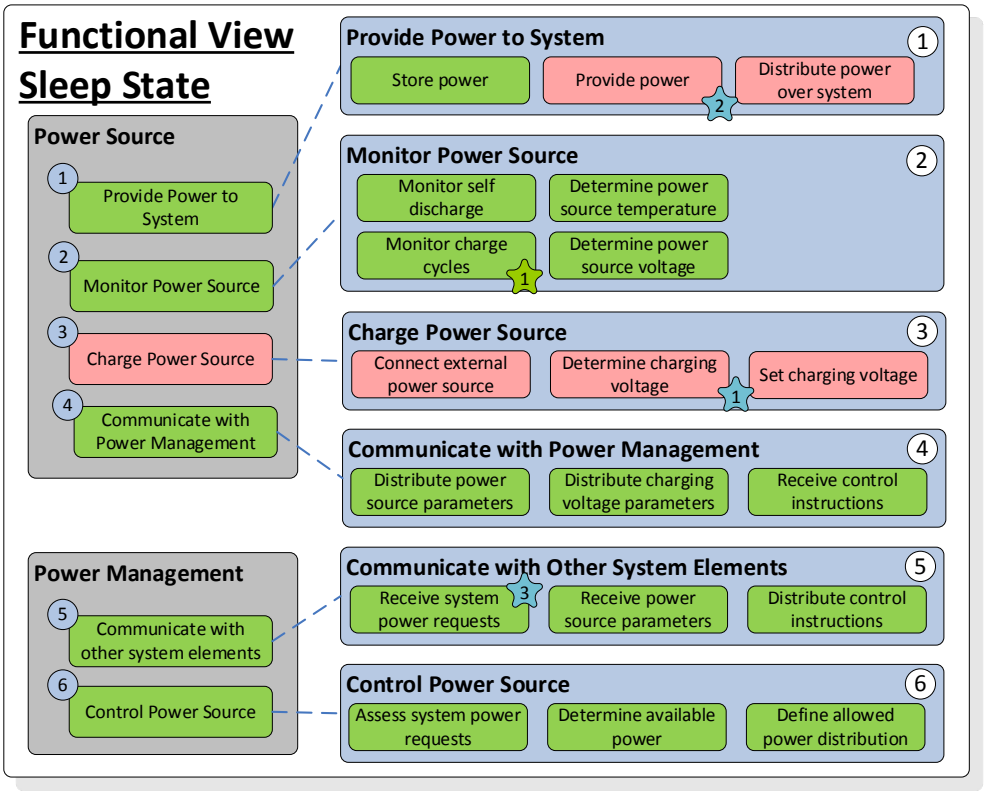


Figure E.7 – Functional View for the Sleep State

E.3 Context Views

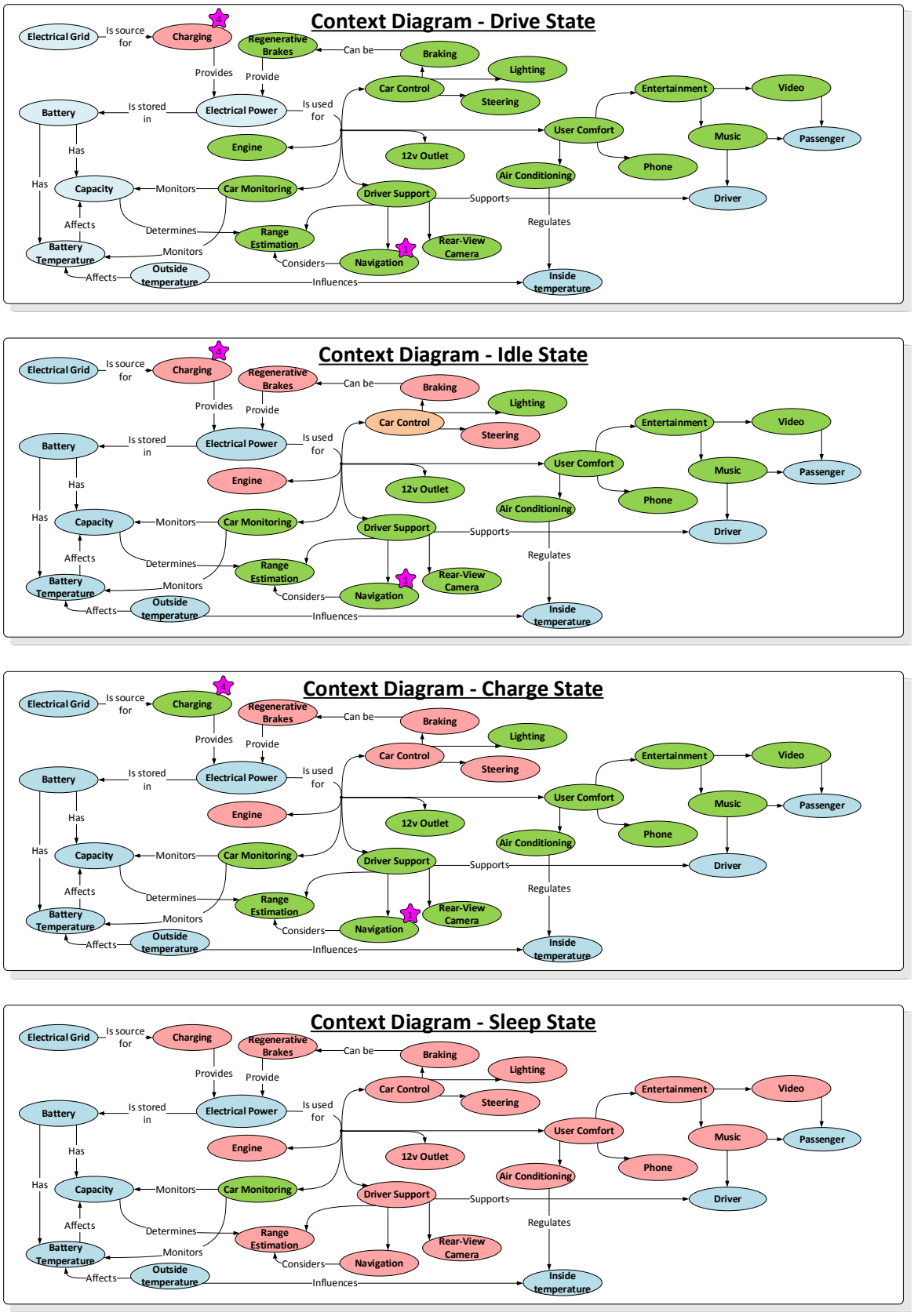


Figure E.8 – Context Diagrams

Appendix F Phobos Sample Return Mission Example

This appendix provides further detail for the Phobos sample return mission example. F.1Section F.1 details the class definitions for the simulation model, while section F.2 characterizes the system model classes for the application and platform elements.

F.1 Categorization of Mission Stages

Table F.1 – Categorization of mission stages into classes (top row) for the application view

Transfers	Maneuvers	Operations
Transfer Earth-Mars	Launch and Direct Escape (Earth)	Deimos Close Proximity Phase
Transfer to Deimos	Re-entry Phase (Earth)	Phobos Close Proximity Phase
Transfer to Phobos	Descent and Landing Phase (Phobos)	Surface Operations Phase
Transfer Mars-Earth	Ascent Phase (Phobos)	
	Departure Phase (Phobos)	

F.2 Characterization of System Model Classes

Table F.2 – Characterization of Transfer class for application view

Name	Explanation or Specification
Type	**Parameter Unit**
ID	Unique identifier for the transfer
String	Unique Name of transfer
Transfer Type	Depending on transfer type, specific launch windows can be determined. Also, for example Hohmann transfers require high thrust
String	Options: Hohmann (Short) / Hohmann (Long) / Other
Departure	Specifies departure point for transfer
String	Options: Earth / Mars / Deimos / Phobos
Arrival	Specifies arrival point for transfer
String	Options: Earth / Mars / Deimos / Phobos
Launch Period Close	If a transfer is a Hohmann transfer, a launch window needs to be selected. The date selected is the date of the launch window closing
Date	Date of launch window closing
Delta-V budget	An estimation of the Delta-V budget for the transfer
Integer	m/s – with uncertainty %
Duration	An estimation of the duration
Integer	Number of days
Mapping	Name of platform instance to which application instance is mapped
String	Options: None + List with names of platform instances

Table F.3 – Characterization of Maneuver class for application view

Name	**Explanation or Specification**
Type	***Parameter Unit***
ID	Unique identifier for the maneuver
String	*Unique Name of maneuver*
Thrust Type	Indicates required type of platform component by specifying the type of thrust
String	*Options: High / Low / None*
Duration	An estimation of the duration is used. More complex models are possible.
Integer	*Number of days*
Delta-V budget	An estimation of the Delta-V budget for the maneuver
Integer	*m/s – with uncertainty %*
Mapping	Name of platform instance to which application instance is mapped
String	*Options: None + List with names of platform instances*

Table F.4 – Characterization of Operation class for application view

Name	**Explanation or Specification**
Type	***Parameter Unit***
ID	Unique identifier for the operation
String	*Unique Name of operation*
Thrust Type	Indicates required type of platform component by specifying the type of thrust
String	*Options: High / Low / None*
Duration	An estimation of the duration is used. More complex models are possible.
Integer	*Number of days*
Delta-V budget	An estimation of the Delta-V budget for the operations
Integer	*m/s – with uncertainty %*
Mapping	Name of platform instance to which application instance is mapped
String	*Options: None + List with names of platform instances*

Table F.5 – Characterization of the Spacecraft element class for the platform view

Name	**Explanation or Specification**
Type	***Parameter Unit***
Name	Unique identifier for the spacecraft element
String	*Unique Name of spacecraft element*
Spacecraft Element (Type)	Some characteristics can be defined based on the type of spacecraft element. For example a capsule does not have thrust (mainly used for re-entry) and a Launcher does not count for weight and delta-V budgets
String	*Options: Launcher / Propulsion Module / Spacecraft / Capsule*
Thrust Type	Indicates the available thrust power
String	*Options: High / Low / High & Low / None*
I_{sp}	Indicates the available specific impulse, which relies on engine design and propellant type
Integer	*Specific impulse in seconds*
Dry Mass	An estimation of the mass of the element excluding fuel
Integer	*Mass (kg) – with uncertainty %*

Appendix G Validation Survey

This survey was designed to have a formalized way of collecting feedback of individuals that have been involved in the creation or usage of interactive system overviews. It has been sent out to five persons within the Philips iXR healthcare division, of which four have responded to the survey.

G.1 Survey Explanation

The survey consisted of 2 introductory and 18 multiple choice questions. Each multiple choice question is a statement for which a qualification of strongly agree - agree – disagree - strongly disagree or don't know can be given. The purpose of this survey was mainly to do a qualitative and not a quantitative assessment of the interactive A3. Therefore, additional comments were requested for each of the questions as well.

G.2 Survey Introduction

The following text was presented to respondents:
This is a survey designed to support the validation of my PhD research in the Allegio project on communication of system behavior using interactive A3 Architecture Overviews. While A3 Architecture Overviews (collecting architecture information and presenting it in an A3 format) have been researched before, they have not been applied to communicate models and simulations. This is where interactive, digital A3 Architecture Overviews come in. Within Philips iXR, an interactive A3 was developed for Power Management redesign, which can be seen being used on an A3 sized tablet in the figure below (omitted in this thesis for confidentiality reasons). For reference, an A3 sized screenshot of the interactive A3 is included at the end of this survey (omitted in this thesis for confidentiality reasons).
Some terminology:
***View** = A single block or group in the A3 Architecture Overview*
***Interactive A3** = A user interface embedded in an A3 Architecture Overview*
***Simulation Model** = The underlying model that determines the transitions and behaviors possible in the interactive A3*

G.3 Survey Results

The survey questions and results are presented in Table G.1, while additional comments given by the survey respondents are presented in

Table G.2.

Table G.1 – Survey Questions and Results

Question 1	**What is your role in system development?**								
System Architect	50%	Domain Specialist (Engineer)	50%	Other	0				
Question 2	**What is your role in system development?**								
0-5 Years	25%	5-10 Years	25%	10+ Years	50%				
Question 3	**I think that simulation models are important in conceptual system design**								
Strongly Agree	50%	Agree	50%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 4	**Connecting a simulation model with an A3 is too much work for the added value**								
Strongly Agree	0%	Agree	0%	Disagree	0%	Strongly Disagree	0%	Don't Know	100%
Question 5	**Connecting a simulation model to an interactive A3 seems too complex**								
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	0%	Don't Know	25%
Question 6	**The notation used in the interactive A3 is complicated**								
Strongly Agree	0%	Agree	0%	Disagree	100%	Strongly Disagree	0%	Don't Know	0%
Question 7	**The interactive A3 size is a problem for its use**								
Strongly Agree	0%	Agree	0%	Disagree	50%	Strongly Disagree	25%	Don't Know	25%
Question 8	**I like the idea of having different views within the same A3**								
Strongly Agree	0%	Agree	100%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 9	**Seeing changes in the system in one single view is helpful**								
Strongly Agree	0%	Agree	100%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 10	**Seeing changes in the system in different views at the same time is helpful**								
Strongly Agree	0%	Agree	100%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 11	**I think an interactive A3 is an appropriate method to explain system behavior**								
Strongly Agree	25%	Agree	50%	Disagree	0%	Strongly Disagree	0%	Don't Know	25%
Question 12	**I think that there are better methods to communicate system behavior**								
Strongly Agree	0%	Agree	0%	Disagree	0%	Strongly Disagree	0%	Don't Know	100%
Question 13	**The interactive A3 is a useful addition for conceptual design**								
Strongly Agree	0%	Agree	75%	Disagree	0%	Strongly Disagree	0%	Don't Know	25%
Question 14	**I expect the interactive A3 to improve communication within my discipline**								
Strongly Agree	25%	Agree	75%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 15	**I expect the interactive A3 to improve communication outside of my discipline**								
Strongly Agree	25%	Agree	75%	Disagree	0%	Strongly Disagree	0%	Don't Know	0%
Question 16	**It is easier to communicate using a report and regular presentations**								
Strongly Agree	0%	Agree	0%	Disagree	25%	Strongly Disagree	0%	Don't Know	75%
Question 17	**It is more valuable to communicate using a report and regular presentations**								
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	0%	Don't Know	25%
Question 18	**This approach can only be successful if it is used in specific cases**								
Strongly Agree	0%	Agree	25%	Disagree	50%	Strongly Disagree	0%	Don't Know	25%
Question 19	**This approach can only be successful if it is implemented company-wide**								
Strongly Agree	0%	Agree	0%	Disagree	75%	Strongly Disagree	25%	Don't Know	0%
Question 20	**I would like to use this approach in my further work**								
Strongly Agree	0%	Agree	75%	Disagree	0%	Strongly Disagree	0%	Don't Know	25%

Table G.2 – Excerpt of comments in survey (SA = System Architect, DS = Domain Specialist)

Question 3	**I think that simulation models are important in conceptual system design**
(SA) Are becoming a must to manage complexity however the current way of working is a long ways from using simulation models effectively	
(DS) Given the complexity of our products, there is little room for making the wrong decision and realizing once it is implemented. By having a good model that can help understand implications with the various stakeholders, the alignment and decision making is done at the beginning of the project when there is more freedom that at the end	
Question 4	**Connecting a simulation model with an A3 is too much work for the added value**
(SA) It does seem like a lot of work to a non-software designer.	
Question 5	**Connecting a simulation model to an interactive A3 seems too complex**
(DS) Probably once the A3 structure is used by the team, the interactive part is needed and appreciated. What seems complex is the "thinking" needed to carry this activity out, but that should be already done regardless of the tool used.	
Question 9	**Seeing changes in the system in one single view is helpful**
(DS) When complexity is low I want to have it all in one view	
Question 10	**Seeing changes in the system in different views at the same time is helpful**
(DS) When complexity is getting higher I want to have the option of having more views	
Question 12	**I think that there are better methods to communicate system behavior**
(SA) Not sure -> I think it is part of communicating system behavior but most times it requires support (aka Power Point or Technical Concept) depending audience.	
Question 14	**I expect the interactive A3 to improve communication within my discipline**
(SA) A good interactive model can always be used to explain that aspect of the system to people less involved with it. The interactive part makes it better to understand.	
Question 17	**It is more valuable to communicate using a report and regular presentations**
(SA) Pending audience -> if concept of A3 is not familiar and audience is not used in conceptual thinking the information requires introduction + good storyline and A3 may not be suitable	
Question 20	**I would like to use this approach in my further work**
(SA) I am using A3 sheets already and they are very helpful -> glad to try the interactive variant	
Overall Remarks	
(DS) This research ambitiously tries to present an alternative to the ways people (at least at my level) are used to communicate. So it will inevitable encounter resistance. The approach of using actual examples and work them out thoroughly is very useful to help understand the value of this tool. I liked the tool and approach and will find a way to use it again.	

Appendix H Full-size A3 Architecture Overviews

This appendix contains full size versions of all A3 Architecture Overviews that are presented throughout this thesis.

H.1 Philips Interventional X-ray Imaging Chain: Latency & Jitter A3 Architecture Overview

Legend

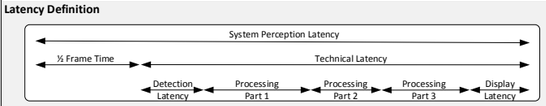
Get X-Ray Image — Function
FD20 Detector — Resource
½ Frame Time — Process
16⅔ Process Latency (ms)
1 Resource Speed (unitless)
30 FPS Clock Period
Conclusion reference

Goals of Modelling Activity

1. Be aware of influence of various system parameters on jitter
2. Align testing procedures with respect to jitter
3. Ensure appropriate design effort on jitter reduction

Definitions

Latency Definition



Jitter is variation in latency. Too much jitter can lead to a glitch (noticeable disturbance for the physician)

Glitch Definition

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur

Relevant Requirements

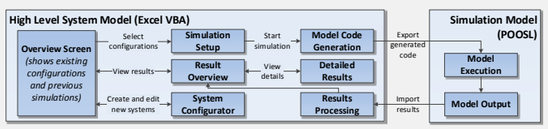
Latency:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur

Imaging Chain: Jitter & Glitches

Author: Steven Haveman (s.haveman@utwente.nl)
Contributors: John Doe, John Smith, Nick Cage, Robert de Vito

Public Version 3-11-2014 v1.0
All data and the system model structure are fictional for confidentiality reasons

Simulation Tool / Model



System model can be configured with various parameters and is simulated using a simulation model written in the POOSL Language. The configuration of systems and the analysis of results is done in Microsoft Excel.

Sources of Jitter

1 Hiccups (hanging processes)

Example PC2 Process	
Task Name	Execution Time (µs)
Example Process 1	4732
Example Process 2	4690
Example Process 3	4610
Example Process 4	4593
Example Process 5	**22416**
Example Process 6	**24209**
Example Process 7	**23580**
Example Process 8	**20005**
Example Process 9	4822
Example Process 10	4738
Example Process 11	4687
Example Process 12	4688

Can be caused by Windows hanging, etc.

2 Variance in process execution times

Measured System Task On PC2: Example Process 2	
Average Execution Time (ms)	Average Execution Time Without Hiccups (ms)
30,1	28,42
Standard Deviation (ms)	Average Standard Deviation Without Hiccups (ms)
3,66	0,4
Variance (%)	Variance (%)
12%	1,4%

Natural / generic variance in execution times

3 System clocks in display components

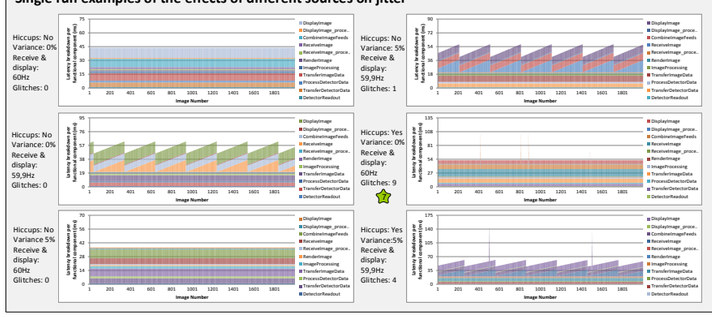
System Clocks	Frequency Range Min	Frequency Range Max	Period (ms) Min	Period (ms) Max
Read Detector	30 FPS	30 FPS	33,33	33,33
Receive Image	59 Hz	61 Hz	16,949	16,393
Display Image	59 Hz	61 Hz	16,949	16,393

Values used in simu.	Frequency	Period
Read Detector	30 FPS	33,3333
Receive Image	59,9Hz	16,6945
Display Image	59,9Hz	16,6945

Periods are not common divisors, thus phase shift occurs

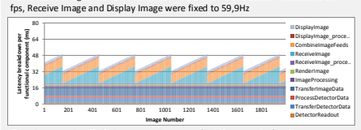
Simulation Results

Single run examples of the effects of different sources on jitter

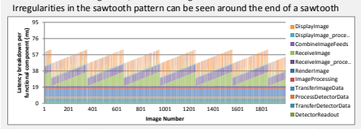


Some runs have extreme amount of glitches (outlier case)

Run of 2000 images with 10% variance in PC-2 and PC-3. Detector Readout is 30 fps, Receive Image and Display Image were fixed to 59,9Hz



The above run has 7 glitches, around average for this amount of variance. Irregularities in the sawtooth pattern can be seen around the end of a sawtooth

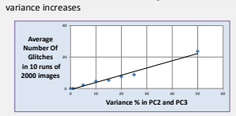


The above run has 176 glitches, much more than average this happened in 2 out of 10 runs at 10% variation. (We call this an outlier case)

Exploring effect of various variances in PC-2 and PC-3 on glitches

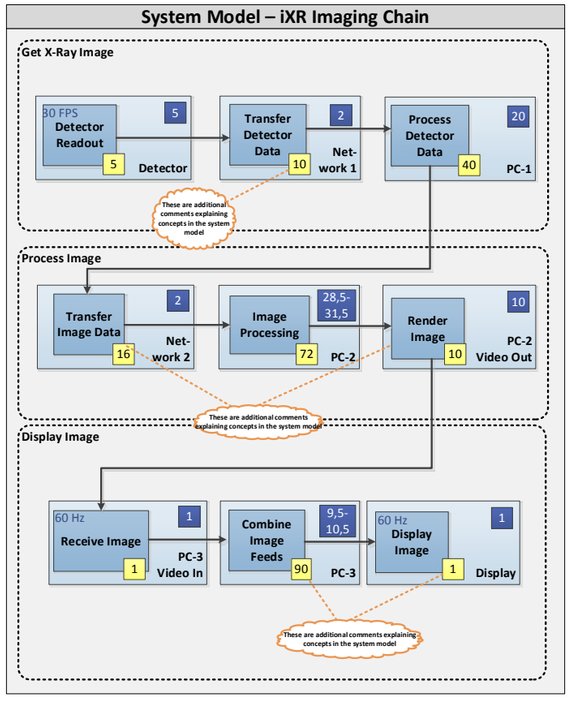
Variance	Average number of glitches in 10 runs of 2000 images
0%	0
1%	0
5%	2,2
10%	38,7 (4,6 without the 3 outlier cases)
15%	26,8 (5,3 without the 2 outlier cases)
25%	47,2 (7,6 without the 2 outlier cases)
25%	35,1 (8,8 without the 2 outlier cases)
50%	73,5 (25,7 without the 3 outlier cases)

This table shows that the number of glitches increases as variance increases



In the graph the averages without outliers are mapped. A linear relation ship can be seen.

System Model – iXR Imaging Chain



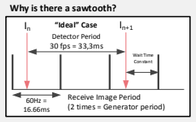
Analysis

What is a glitch?

Explanatory Visual Aid (omitted for confidentiality)

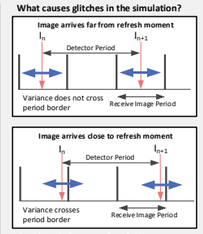
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

Why is there a sawtooth?



As the receive image period is larger, latency will increase up to the point that there are not 2 but only 1 period between images

What causes glitches in the simulation?



When the wait time nears the refresh moment an image can end up on either side. Over time, we move out this "danger zone"

No variance in execution times leads to no overlap in sawtooths

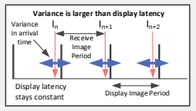
With variance in execution times, sawtooths overlap

As near the border, images can end up in both of the sawtooths

This leads to the above behaviour which in turn leads to glitches

What causes the outlier cases?

If the refresh time of the **display** is around the refresh time of the **image grabber** plus **the average time it takes for an image to get from the image grabber to the display**, the following occurs:



So, when we have two subsequent clocks have a similar period, we stay in the border area all the time with a unfavorable start time. This leads to a lot of glitches

Validation – What do the simulation results mean for the actual system

What is the actual variance and framerate?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

Explanatory Table (omitted for confidentiality)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

Explanatory Table (omitted for confidentiality)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

What is the actual chance on outliers (extreme glitch cases)?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

Explanatory Graph (omitted for confidentiality)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

Explanatory Graph (omitted for confidentiality)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

A closer look at glitch definition

Definition 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Definition 2
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Current measures used in system to control glitches

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris

Explanatory Visual Aid (omitted for confidentiality)

Conclusion & Recommendations

Main Observations / Conclusions

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusm
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusm
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
4. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
5. Lorem ipsum dolor sit amet, consectetur adipiscing elit
   - Lorem ipsum dolor sit amet, consectetur adipiscing elit
   - Lorem ipsum dolor sit amet, consectetur adipiscing elit
   - Lorem ipsum dolor sit amet, consectetur adipiscing elit
6. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
7. Lorem ipsum dolor sit amet, consectetur adipiscing elit

Recommended glitch definition to use across all specifications
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatu

Possible future measures to alleviate glitches from higher variance

This behaviour causes glitches

Add delay for early images

Results in shorter sawtooth and slightly higher latency

Figure H.1 – Latency & Jitter A3 used to present and communicate results of behavioral analysis of hand-eye coordination in the iXR imaging chain (abstractions have been made for confidentiality)

H.2 Philips Interventional X-ray Power Management Subsystem

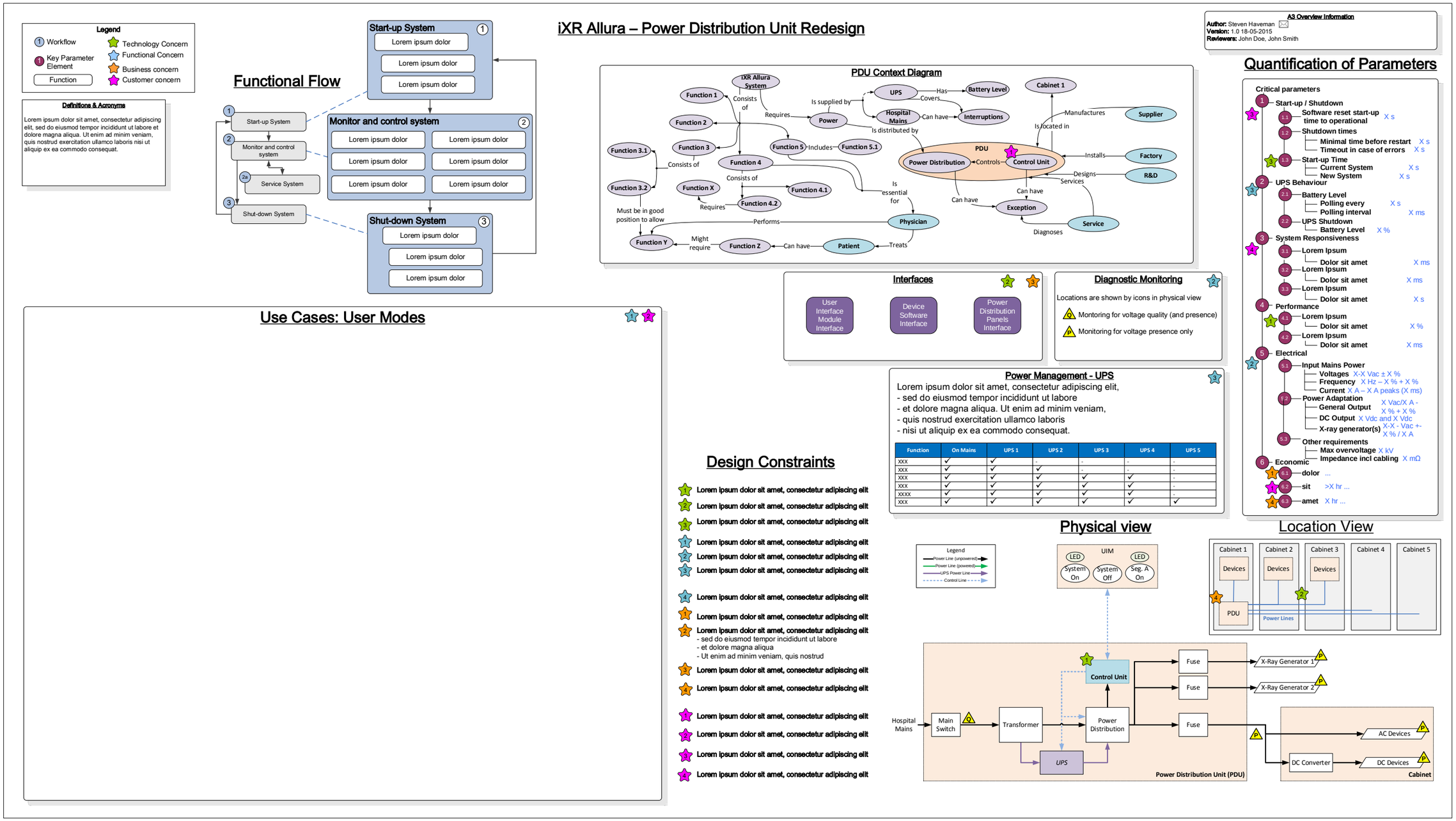


Figure H.2 – A3 Architecture Overview for Philips iXR power management subsystem used as basis for interactive version (abstractions have been made for confidentiality)

H.3 Electric Car – Power Management Example

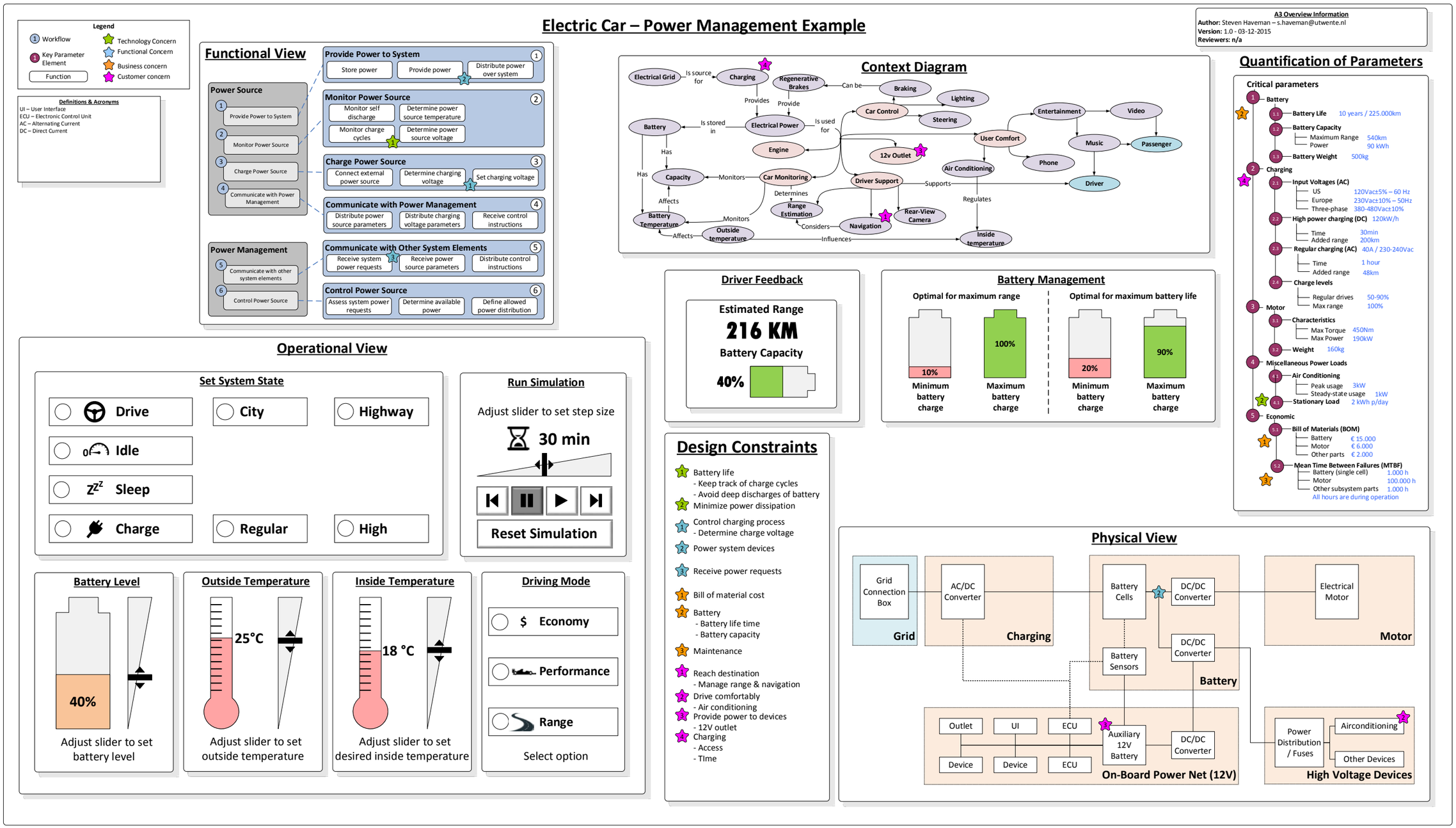


Figure H.3 – Interactive system overview used to communicate system behavior of the power management subsystem

H.4 Phobos Sample Return Mission Timeline – Structural Views

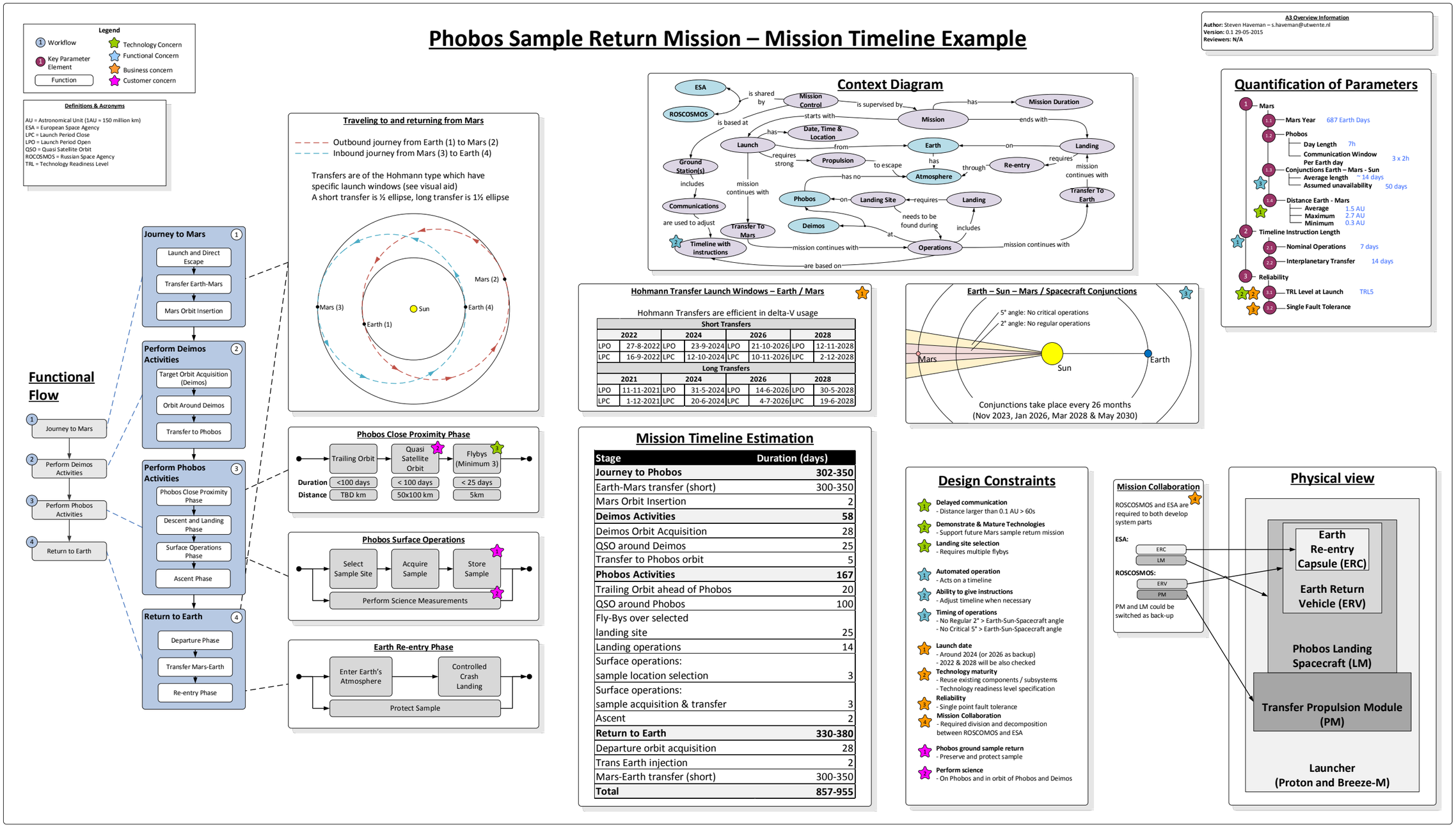


Figure H.4 – Initial structural views used to frame mission timeline concern for the Phobos sample return mission

H.5 Phobos Sample Return Mission Timeline– Behavioral Views

Legend

- Workflow
- Key Parameter Element
- Function
- Technology Concern
- Functional Concern
- Business concern
- Customer concern

Definitions & Acronyms

AU = Astronomical Unit (1AU ≈ 150 million km)
ESA = European Space Agency
LPC = Launch Period Close
LPO = Launch Period Open
QSO = Quasi Satellite Orbit
ROCOSMOS = Russian Space Agency
TRL = Technology Readiness Level

System Definition Symbols

- TR: Mission Stage Class: Transfer
- OP: Mission Stage Class: Operations
- M: Mission Stage Class: Manoeuvre
- Weight in kilograms
- Isp: Specific impulse in seconds
- ΔV: Delta-V in m/s
- DAYS: Duration in days

Functional (Application) View

	Application Element	ΔV	DAYS	Mapping
TR	Launch and Direct Escape	0,0	24/12/10	Launcher (Proton and Breeze-M)
M	Transfer Earth-Mars	73,5	332	Transfer Propulsion Module (PM)
M	Mars Orbit Insertion	864,2	2	Transfer Propulsion Module (PM)
M	Target Orbit Acquisition (Deimos)	674,1	28	Transfer Propulsion Module (PM)
OP	Orbit Around Deimos	50,4	25	Transfer Propulsion Module (PM)
M	Transfer to Phobos	840,0	5	Transfer Propulsion Module (PM)
O	Phobos Close Proximity Phase	168,0	145	Phobos Landing Spacecraft (LM)
M	Descent and Landing Phase	165,0	14	Phobos Landing Spacecraft (LM)
O	Surface Operations Phase	312,5	6	Phobos Landing Spacecraft (LM)
M	Ascent Phase	36,0	2	Earth Return Vehicle (ERV)
M	Departure Orbit Acquisition	898,8	30	Earth Return Vehicle (ERV)
TR	Trans Earth Injection	839,0	26/08/03	Earth Return Vehicle (ERV)
M	Transfer Mars-Earth	73,5	320	Earth Return Vehicle (ERV)
M	Re-entry Phase	0,0	0	Earth Re-entry Capsule (ERC)

Phobos Sample Return Mission – Mission Timeline Example

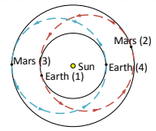
A3 Overview Information
Author: Steven Haveman – s.haveman@utwente.nl
Version: 0.1 29-05-2015
Reviewers: N/A

Main Mission Timeline Influences

Mission Stage Duration
Stage lengths together determine total mission timeline. However, variations are possible

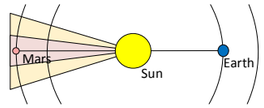
Choices in science to be done influences those stage lengths
E.g. Deimos Science:
- None (drop stage)
- Fly-By (short)
- QSO (long)

Launch Windows
Optimal transfers with respect to time and delta-V are Hohmann



Hohmann transfers have specific launch windows

Conjunctions
Limited communication possible during Earth-Sun-Mars alignment



Conjunctions taking place during a critical mission stage near Mars must be avoided

Design Constraints

- **Delayed communication** - Distance larger than 0.1 AU > 60s
- **Demonstrate & Mature Technologies** - Support future Mars sample return mission
- **Landing site selection** - Requires multiple flybys
- **Automated operation** - Acts on a timeline
- **Ability to give instructions** - Adjust timeline when necessary
- **Timing of operations** - No Regular 2° > Earth-Sun-Spacecraft angle - No Critical 5° > Earth-Sun-Spacecraft angle
- **Launch date** - Around 2024 (or 2026 as backup) - 2022 & 2028 will be also checked
- **Technology maturity** - Reuse existing components / subsystems - Technology readiness level specification
- **Reliability** - Single point fault tolerance
- **Mission Collaboration** - Required division and decomposition between ROSCOMOS and ESA
- **Phobos ground sample return** - Preserve and protect sample
- **Perform science** - On Phobos and in orbit of Phobos and Deimos

Quantification of Parameters

1 Mars
- 1.1 Mars Year: 687 Earth Days
- 1.2 Phobos
  - Day Length: 7h
  - Communication Window Per Earth day: 3 x 2h
- 1.3 Conjunctions Earth – Mars - Sun
  - Average length: ~ 14 days
  - Assumed unavailability: 50 days
- 1.4 Distance Earth - Mars
  - Average: 1.5 AU
  - Maximum: 2.7 AU
  - Minimum: 0.3 AU

2 Timeline Instruction Length
- 2.1 Nominal Operations: 7 days
- 2.2 Interplanetary Transfer: 14 days

3 Reliability
- 3.1 TRL Level at Launch: TRL5
- 3.2 Single Fault Tolerance

Mission Timeline Estimation

Mission Stage	Start Date	End Date	Days	Remarks
Launch and Direct Escape	12-10-2024			
Earth-Mars transfer (short)	12-10-2024	9-9-2025	332	
Mars Orbit Insertion	9-9-2025	11-9-2025	2	
Deimos Orbit Acquisition	11-9-2025	9-10-2025	28	
Deimos Close Proximity Operations	9-10-2025	3-11-2025	25	
Transfer to Phobos orbit	3-11-2025	8-11-2025	5	
Phobos Close Proximity Operations	8-11-2025	22-5-2026	195	50 days extra due to conjunction
Phobos Descent and Landing	22-5-2026	5-6-2026	14	
Phobos Surface Operations	5-6-2026	11-6-2026	6	
Ascent	11-6-2026	13-6-2026	2	
Departure orbit acquisition	13-6-2026	13-7-2026	30	
Trans Earth Injection	3-8-2026			21 days margin before transfer
Mars-Earth transfer (short)	3-8-2026	19-6-2027	320	
Earth Re-Entry	19-6-2027	19-6-2027	0	
Total	**12-10-2024**	**19-6-2027**	**980 days**	**2,7 years**

Hohmann Transfer Launch Windows – Earth / Mars

Hohmann Transfers are efficient in delta-V usage

	Short Transfers Earth-Mars				Long Transfers Earth-Mars			
	2022	2024	2026	2028	2022	2024	2026	2028
LPC	16-9-2022	12-10-2024	10-11-2026	2-12-2028	1-12-2021	20-6-2024	4-7-2026	19-6-2028
Arrival	7-9-2023	9-9-2025	9-9-2027	26-9-2029	31-1-2024	1-8-2026	13-7-2028	6-5-2030
Days	356	332	303	298	791	772	740	686

	Short Transfers Mars-Earth				
	2024	2026	2028	2030	2032
LPC	22-7-2024	3-8-2026	6-9-2028	9-11-2030	5-2-2033
Arrival	6-4-2025	19-6-2027	11-8-2029	22-9-2031	21-10-2033
Days	320	320	339	317	320

Mission Collaboration

ROSCOSMOS and ESA are required to both develop system parts

ESA: ERC, LM
ROSCOSMOS: ERV, PM

PM and LM could be switched as back-up

Wet Mass Calculations

Total Delta-V	Dry Mass	Fuel Mass
0 m/s	35,0 kg	0,0 kg
1847,25 m/s	220,0 kg	258,8 kg
645,5 m/s	721,1 kg	445,9 kg
2428,65 m/s	674,0 kg	2675,5 kg

Total Wet Mass
5156,2 kg

Physical (Platform) View

- Earth Re-entry Capsule (ERC): 35,0 kg; N/A
- Earth Return Vehicle (ERV): 220,0 kg; 302
- Phobos Landing Spacecraft (LM): 721,1 kg; 220
- Transfer Propulsion Module (PM): 674,0 kg; 332
- Launcher (Proton and Breeze-M): 100,0 kg; N/A

Figure H.5 – Interactive system overview used to communicate system behavior of the Phobos sample return mission timeline