Downloaded from orbit.dtu.dk on: Dec 20, 2017

Technical University of Denmark

Test-Driven, Model-Based Systems Engineering.

Munck, Allan; Madsen, Jan; Lindqvist, Lars; Schmidt, Flemming ; Pop, Paul

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Munck, A., Madsen, J., Lindqvist, L., Schmidt, F., & Pop, P. (2017). Test-Driven, Model-Based Systems Engineering. DTU Compute. (DTU Compute PHD-2017, Vol. 444).

DTU Library
Technical Information Center of Denmark

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Test-Driven, Model-Based Systems Engineering

Allan Munck

Kongens Lyngby 2017
PhD-2017-444

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk

PhD-2017-444
ISSN 0909-3192

Summary (English)

Hearing systems have evolved over many years from simple mechanical devices (horns) to electronic units consisting of microphones, amplifiers, analog filters, loudspeakers, batteries, etc. Digital signal processors replaced analog filters to provide better performance end new features. Central processors were added to provide many functions for monitoring and controlling other parts of the devices. Hearing systems have thus evolved into complex embedded system.

Radio systems were added to allow hearing aids to communicate with accessories, auxiliary equipment, third-party products, etc. Many new features are enabled by such radio communication. Monitoring and controlling hearing aids from remote control devices or smart phones have been incorporated into several products. Direct audio streaming between hearing aids and dedicated streaming devices or smart phones is possible with some products. Also emerging are advanced features that are based on interactions with internet services, clouds, etc. Hearing systems are thus evolving into large and complex smart systems.

Designing complex embedded systems or large smart systems are notoriously difficult. Many systems are still developed using document-based methods, where requirements and proposed architecture are described textually with the addition of a few figures and tables. Such documents cannot be subjected to testing, so it is impossible to predict the functionality and performance or even feasibility of the intended systems.

Replacing documents with models have several advantages. Models can be simulated and analyzed such that functionality and performance can be predicted before any parts have been built. Potential flaws in the specification can therefore be corrected in early phases, which may reduce development effort and costs.

This thesis concerns methods for identifying, selecting and implementing tools for various aspects of model-based systems engineering. A comprehensive method was proposed that include several novel steps such as techniques for analyzing the gap between requirements and tool capabilities. The method was verified with good results in two case studies for selection of a traceability tool (single-tool scenario) and a set of modeling tools (multi-tool scenarios).

Models must be subjected to testing to allow engineers to predict functionality and performance of systems. Test-first strategies are known to produce good results in software development. This thesis concerns methods for test-driven modeling of hearing systems.

A method is proposed for test-driven modeling of embedded systems of medium complexity. It utilizes formal model checking to guarantee functionality and performance. Test-driven design space exploration is enabled by using statistical model checking to obtain estimates that are verified formally at the final stages of the method. The method was applied with good results to a case study, where two solutions to a design problem were developed and verified. Feasible ranges for critical parameters were identified. Both solution conformed to all requirements.

Smart systems are typically too large and complex to be verified by formal model checking, and the research showed that statistical model checking in its current form cannot be used for verifying such systems. A new method is therefore proposed for test-driven modeling of smart systems. The method uses formal verification of basic interactions. Simulations are used for verifying the overall system. To predict performance for scenarios that are too large to be simulated, the method uses mathematical forecasting based on simulating series of smaller scenarios, fitting simulation results to estimator functions, and extrapolating beyond the simulated data set. Mathematical forecasting allowed us to predict the performance of system scenarios that were much too large to be simulated. Such performance estimates may be somewhat imprecise but are nevertheless valuable because they provide answers that cannot be obtained otherwise.

The research has thus proposed and verified methods for selecting modeling tools and for test-driven systems modeling for the benefit of GN Hearing and other organizations involved in development of complex embedded systems of large smart systems.

Summary (Danish)

Høresystemer har udviklet sig igennem mange år fra simple mekaniske anordninger (horn) til elektroniske apparater bestående af mikrofoner, forstærkere, analoge filtre, højttalere, batterier, osv. Digitale signalprocessorer har erstattet analoge filtre for at give bedre og mere avanceret lydbehandling. Centrale processorer blev tilføjet for at gøre det muligt at overvåge og styre andre dele af apparaterne. Høresystemer har således udviklet sig til komplekse indlejrede systemer.

Radiosystemer blev tilføjet, så høreapparater kan kommunikerer med tilbehør, extraudstyr, tredjeparts produkter, etc. Mange nye funktioner bliver mulige med sådanne radio systemer. Overvågning og styring af høreapparater fra fjernbetjeningsenheder eller smartphones er allerede indarbejdet i flere produkter. Direkte audio-streaming mellem høreapparater og dedikerede streaming-enheder eller smartphones er muligt med nogle produkter. Også nye avancerede funktioner, der er baseret på interaktioner med internettjenester, clouds, osv., er ved at dukke op. Høresystemer er således ved at udvikle sig til store og komplekse smart-systemer.

Design af komplekse indlejrede systemer eller store smart-systemer er notorisk vanskeligt. Mange systemer bliver stadig udviklet ved hjælp af dokumentbaserede metoder, hvor krav og foreslået arkitektur er beskrevet med tekst og tilhørende figurer og tabeller. Sådanne dokumenter kan ikke udsættes for testafvikling, hvorfor det er umuligt at forudsige funktionalitet og ydeevne, eller om de påtænkte systemer er gennemførlige.

Udskiftning af dokumenter med modeller har flere fordele. Modeller kan simuleres og analyseres således, at funktionalitet og ydeevne kan forudsiges, før systemernes komponenter udvikles og fremstilles. Potentielle fejl i specifikationen kan derfor rettes i tidlige faser, hvilket kan reducere udviklingsarbejdet og omkostninger.

Denne afhandling omhandler metoder til at identificere, udvælge og indføre værktøjer til forskellige aspekter af modelbaseret system-ingeniørarbejde. En omfattende metode, der inkluderer flere nye elementer såsom teknikker til at analyse gabet mellem behov og værktøjers formåen, er blevet foreslået. Metoden blev verificeret med gode resultater i to studier for udvælgelse af et sporbarhedsværktøj (scenarie med et enkelt værktøj) og et sæt af modelleringsværktøjer (scenarie med flere værktøjer).

Modeller skal testes for at tillade ingeniører at forudsige systemers funktionalitet og ydeevne. Test-først strategier er kendt for at give gode resultater i softwareudvikling. Denne afhandling omhandler metoder til testdrevet modellering af høresystemer.

En metode til testdrevet modellering af indlejrede systemer af medium kompleksitet er blevet foreslået. Metoden benytter formel verifikation af modeller til at garantere funktionalitet og ydeevne. Testdrevet udvikling af designalternativer er muliggjort ved hjælp af statistisk verifikation af modeller, hvor estimater efterfølgende verificeres formelt. Metoden blev brugt med gode resultater i et studie, hvor to forskellige løsninger til et designproblem blev udviklet og verificeret. Mulige intervaller for kritiske parametre blev identificeret. Begge løsninger opfyldte alle krav.

Smart-systemer er typisk for store og komplekse til at blive verificeret med formel verifikation, og den gennemførte forskning viste, at statistisk verifikation i sin nuværende form ikke er egnet til verifikation af sådanne systemer. En ny metode til testdrevet modellering af smart-systemer er derfor blevet foreslået. Metoden bruger formel verifikation af basale interaktioner. Simuleringer bruges til at verificere det samlede system. For at kunne forudsige systemers ydeevne for scenarier, der er for store til at blive simuleret, benytter metoden matematiske prognoser baseret på simulering af en række mindre scenarier, udledning af estimeringsfunktioner fra simuleringsresultaterne, og extrapolering udover det simulerede datasæt. Matematiske prognoser tillod os at forudsige ydeevnen for scenarier, der var for store til at blive simuleret. Sådanne estimater for systemers ydeevne kan være noget upræcise, men ikke desto mindre værdifulde, fordi de giver svar, som ikke kan opnås på anden vis.

Forskningen har således foreslået og verificeret metoder til udvælgelse modelleringsværktøjer og for testdrevet systemmodellering til gavn for GN Hearing og andre organisationer, der udvikler komplekse indlejrede systemer eller store smart-systemer.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a PhD degree.

The thesis deals with the application of test-driven, model-based systems engineering for developing advanced hearing systems in a medical device company, GN Hearing.

Professor Jan Madsen from DTU Compute and Senior Vice President Lars Lindqvist from GN Hearing supervised the work. Professor Paul Pop from DTU Compute and Systems Engineering Architecture Manager Flemming Schmidt from GN Hearing co-supervised the work. Director of Systems Engineering Michael Woeroesch from GN Hearing provided invaluable support for completing the project.

Lyngby, 28-February-2017

Allan Munck

Allan Munck

Papers

The papers included in this thesis are:

- Allan Munck and Jan Madsen, *Test-driven modeling of embedded systems.* Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC), 2015. IEEE, 2015. Published.
- Allan Munck and Jan Madsen, *A systematic and practical method for selecting systems engineering tools.* Accepted for presentation at 11th Annual IEEE International Systems Conference, 2017. To be published in the conference proceedings in IEEE IEL and IEEE Xplore.
- Allan Munck and Jan Madsen, *Test-driven modeling and development of cloud-enabled cyber-physical smart systems.* Accepted for presentation at 11th Annual IEEE International Systems Conference, 2017. To be published in the conference proceedings in IEEE IEL and IEEE Xplore.

Acknowledgments

I would like to thank my primary supervisors, Professor Jan Madsen from DTU Compute and Senior Vice President Lars Lindqvist from GN Hearing, for initiating, funding and supporting the project throughout all phases. Especially I would like to thank Jan Madsen for his continuous involvement, tireless optimism, help and support.

I would also like to thank my co-supervisors, Professor Paul Pop from DTU Compute and Systems Engineering Architecture Manager Flemming Schmidt from GN Hearing for help, support and stimulating discussions. Especially my frequent interactions with Flemming Schmidt have been of great value. I would also like to thank Director of Systems Engineering Michael Woeroesch from GN Hearing for his interest and engagement at promoting the completion of the work.

I would like to thank various students and staff members at DTU Compute as well as various co-workers at GN Hearing for inspiring discussions and for their interest in the PhD project.

I would like thank GN Hearing, Innovation Fund Denmark and the Confederation of Danish Industry for funding this PhD project. Finally, I would like to thank all persons involved in the ITOS project under the Confederation of Danish Industry, especially Senior Advisor and Project Leader Henrik Valentin Jensen for his involvement and strong interest in this PhD project.

My studies became extremely interesting due to their involvement, help and support.

Contents

List of Figures

List of Tables

Vocabulary

Abbreviations and terms

20-sim	20sim modeling tool.
AADL	Architecture Analysis and Design Language.
AAM	Architectural Analyses Modeling.
ABCM	Architecture/Behavior Co-Modeling.
ABMS	Agent-Based Modeling and Simulating.
ADL	Architecture Description Languages.
AHP	Analytical Hierarchy Process, tool selection method.
AIS	Artificial Intelligence Systems.
Algo	DSP Algorithm Software.
ALM	Application life-cycle Management.
AMDD	Agile Mode-Driven Development.
AMS	Analogue mixed signal.
API	Application Programming Interface.
APP	Mobile APPlication, in case study.
#AR	Number of All Requirements, in tool selection.
Architecture	Hierarchy, properties and interconnections systems.
ASIC	Application Specific Integrated Circuit

`Assertion`	Logical test that forces execution to create an exception and crash if the test evaluates to false, in simulation.
`Basic interaction`	Set of strongly related messages between entities in simulation model, e.g. transmission and confirmation of a specific type of notification.
`Battry`	Electrical element that supply power to electronic devices.
`BDD`	SysML Block Definition Diagram.
`BDS`	Big-Data systems.
`Bluetooth bridge`	Device that facilitate communication between HIs and mobile devices (smartphones).
`BOM`	Bill Of Materials.
`Bottom-up`	Modeling method where modeling starts at the lowest level of details and progresses by adding elements at increasingly higher levels of abstraction, in contrast to top-down.
`BPEL`	Business Process Execution Language.
`BPMN`	Business Process Modeling Notation.
`BPSS`	Business Process Simulation Software
`BSM`	Behavioral Simulation Modeling.
`BTB`	BlueTooth Bridge device.
`CAE`	Computer Aided Engineering.
`CAPI`	Common API.
`CASE`	Computer Aided Systems Engineering
`CECPS`	Cloud-Enabled Cyber Physical Smart, system.
`Channel`	1: Frequency range, in radio communication. 2: Communication variable, in the UPPAAL modeling language.
`CM`	Coherent modeling.
`cmds`	Commands.
`Code generation`	The activity of generating (simulation) code from models (semi-) automatically.
`Contextual system`	Systems in the system domain the SUD interfaces to.

Co-variation	Varying two or more parameters in a single simulation experiment.
CPS	Cyber-Physical System.
CPU	Central Processing Unit.
Crowd	Large collection of users
CT	Continuous Time, in simulation
Cyber-physical system	System that comprise both physical parts and (internet-based) cyber parts.
D	Desired, requirements or features, in tool selection.
Data fitting	Technique to derive estimator functions that fit data sets as closely as possible (or with sufficient precision).
DBSE	Document Based Systems Engineering.
DE	1: Discrete Event, in simulation. 2: Domain Expert, in SE and modeling.
Deadlock	A situation where two or more processes wait infinitely for each other, in simulation.
Descriptive model	Model that is used for describing (a part of) a system without adding details that allow simulation or other forms of analysis.
DI	Confederation of Danish Industry.
Document-based	Development method that uses documents to describe the intended system.
Domain	See system domain.
Double-conformation	Pattern for basic interaction between two entities to ensure that data are transferred reliably, in simulation.
#DR	Number of "Desired" Requirements, in tool selection.
DSE	Design space Exploration.
DSML	Domain-Specific Modeling Language.
DSP	Digital Signal Processing/Processor.
DTU	Technical University of Denmark.
DTU-Compute	Department of Applied Mathematics and Computer Science at DTU.

E2E	End-2-End.
EA	Enterprise Architecture.
EAF	Enterprise Architecture Framework.
ECSAM	Embedded Computer-based System Analysis Method.
EM	Enterprise Modeling.
Embedded system	Physical system that comprises an embedded computer.
EmbeddedWiki	Project under the Inifinit network.
Emergent behavior	Behavior of a system that (only) emerges by combining the parts of a system, often unintended.
End user	Primary user of systems.
Enterprise modeling	The activity of modeling the entire enterprise of engineering systems.
Entity	Computational unit that represent a user, an environmental system, a sub-system, or a component of a systems; in simulation.
Environmental system	System in the environment that affects the operation of a SOI.
ESE	Embedded Systems Engineering, section at DTU-Compute.
EU	End user, in case study.
Event	Message with associated that are sent from one entity to another, in simulation.
Expert system	Tool for selecting SE and other tools based on collections of data concerning the tool of interest.
Extreme programming	Agile development method.
F	Feature requirements, in tool selection.
FDA	U.S. Food and Drug Administration.
Flash	Electronic memory circuit for non-volatile storage.
FM	1: Frequency Modulation. 2: Fundamental Modeling.
FMC	Formal Model Checking.

FMEA	Failure Modes and Effects Analysis.
Formal models	Models that are described in formal modeling language using precise semantics.
Formal modeling	Representing systems as formal models.
Formal verification	Technique to verify formal models such that all reachable state are analyzed.
FPU	Floating Point Unit, processor.
FSM	Finite State Machine.
FSW	Fitting SoftWare, in case study.
G	General requirements, in tool selection.
Gap analysis	Analysis of the gaps between required and provided functionality, performance, reliability, etc.
GHz	Giga Hertz, unit of frequency
GN	Great Nordic, company.
GNH	GN Hearing, company within GN.
GPU	Graphics Processing Unit
GUI	Graphical User Interface.
Hearing system	A system that includes hearing instruments, accessories, 3rd-party devices, web-services, cloud systems, etc.
HCP	Hearing Care Professional, in case study.
HDF	Heterochronous Data Flow, in simulation.
HE	Hypothesis Evaluation, evaluation of a RH.
Hearing care professional	Person at hearing clinics that fit HIs to end-users.
Heterogeneous system	System that contains elements that require different modes of computation to simulate.
HI	Hearing Instrument.
HKBS	Hybrid Knowledge-based System, expert tool.
Homogeneous system	System where all constituent elements can be modeled and simulated by using only one mode of computation.
HW	Hardware.

IaaS	Infrastructure as a Service.
IBD	SysML Internal Block Diagram.
IE	Integrated Engineering.
IM	IMplementation domain.
INCOSE	The International Council on Systems Engineering.
Infinit	Danish network for innovative utilization of IT.
info	Information.
Invariant	Condition that must hold at all times during simulation.
IoT	Internet of Things.
ITEK	Business community for IT, telecom, electronics and communications under DI.
ITOS	Project under DI ITEK.
LED	Light Emitting Diode.
LTL	Linear temporal Logic.
M	Mandatory, requirements or features, in tool selection.
MARTE	Modeling and Analysis of Real Time and Embedded systems (OMG standard).
Mathematical forecasting	Technique to predict values of variables for scenarios that cannot be simulated, based on data fitting and extrapolation.
MB	1: Mega Byte, unit of memory size. 2: Model Builder, in SE and modeling.
MBSE	Model-Based Systems Engineering.
MBT	Model-Based Testing, see also MDT.
MDA	Model-Driven Architecture, modeling with code generation.
MDD	Model-Driven Development.
MDE	Model-Driven Engineering.
MDT	Model-Driven Testing, see also MBT.
Methodology	Methods applied in engineering.
MHz	Mega Hertz, unit of frequency.

μS	Micro Second, unit of time.
MM	Master model, in coherent modeling.
MoC	Mode of Computation, in simulations.
Mock-up	Simple physical model of device or system.
Model-based	Development method that uses models instead of documents (or as supplement to documents).
Modelica	Modeling tool for heterogeneous systems.
Modeling technologies	Sum of modeling methodologies, languages, and tools.
MOO	Multi-Objective Optimization, in tool selection.
#MR	Number of Mandatory Requirements, in tool selection.
MSC	Message Sequence Charts, in simulation.
Multi-tool scenario	Scenario in tool selection, where a set of complementary tools are selected such that all essential requirements are met.
NFP	Non-Functional Property.
Notification	Message that is sent from one entity to another, in simulation.
O	Optional, requirements or features, in tool selection.
OCL	Object Constraint Language (OMG standard).
OMG	Object Management Group.
OOSEM	Object Oriented System Engineering Method.
#OR	Number of "Optional" Requirements, in tool selection.
OUC	Object Under Consideration.
PaaS	Platform as a Service.
PD	Product Development.
PLM	Product Life-cycle Management
Poll	Feature in mobile applications that contacts service provider regularly for obtaining messages or data.
Post-condition	Condition that must hold after completion of a simulation.

Pre-condition	Condition that must hold before starting a simulation.
PtII	Ptolemy-II.
Ptolemy-II	Modeling and simulating tool for heterogeneous systems.
Push	Feature in mobile devices that allow users to register for automatic pushing of messages from service providers to the mobile application.
Qualitative methods	Method for evaluating based on descriptive analysis of features, in tool selection.
Quantitative methods	Method for evaluating based on quantitative analysis using WSM, AHP or similar techniques, in tool selection.
Query	Expression that represent a formal requirement that most hold for a system model, in UPPAAL.
RA	Research Answer, answer to research question.
Radio system	Part of hearing instruments that facilitate communication between the devices and other parts of the hearing system.
RAM	Random Access Memory.
RC	Remote Control.
R&D	Research and Development.
req	Request.
RFT	Remote Fine Tuning, in case study.
RFP	Remote Fine Tuning Package, in case study.
RH	Research Hypothesis.
RM	Requirements Management
ROOM	Real-time Object Oriented Modeling.
RQ	Research Question
rsp	Response.
RTOS	Real Time Operating System.
SaaS	Software as a Service.
SAS	Stationary Audio Streamer.

Scenario	Sequence of events and activities, in simulation.
Scenario simulation	Simulating a model that implements a specific scenario without exercising all reachable states.
SCF	Source code files.
SCP	Smart-Connected Products/devices.
SD	1: Sequence diagram. 2: Specification Document.
SDF	Synchronous Data Flow, in simulation.
SDS	System Design Specification.
SE	Systems Engineering.
Simulink	Modeling tool from MathWorks.
SimJava2	Simulation framework for DE simulations.
SimTAny	Modeling tool for the TAS method, see also VeriTAS.
Simulation	Execution of models to predict functionality and performance.
Simulation domain	See MoC.
Simulation framework	Extendable simulation software.
Simulation series	Series of simulations where exactly one parameter is changed progressively from one value to another in small steps such that one simulation is executed for each step.
Simulation software	Software tool that allow engineers to simulate models.
Single-tool scenario	Scenario in tool selection, where only one tool must be selected to cover the requirement as good as possible.
Smart system	System that provide smart features that utilizes webservices, cloud solutions, artificial intelligence, etc.
SMC	Statistical model checking.
SM	Spouse microphones, same as WM.
SOI	System of Interest.
SoS	System of Systems.
SPI	Serial Peripheral Interface, bus.

SPL	Sound Pressure Level.
SRD	System Requirements Documents.
State explosion	Exponential growth of system states causing memory exhaustion, in simulation.
Stateflow	Modeling tool from MathWorks.
Statistical model	System models that are described in modeling languages that support statistical verification.
Statistical verification	Method to verify models statistically such that functionality and performance can be guaranteed with specifiable probability.
SUC	System Under Consideration.
SUD	System Under Development.
SW	Software.
SWOT	Strengths, weaknesses, opportunities, and threats; analysis of.
SysML	Systems Modeling Language.
System domain	Operational context of a system and its users and environmental systems.
System of systems	System that includes sub-systems over which the developers of the SOI have no control.
SystemC	Systems modeling language that utilize C++.
Tag	Enumeration value that identify a specific type of message, in simulation.
TAS	Test-Driven Agile Simulation, method.
TCP/IP	Transmission Control Protocol / Internet Protocol.
TCTL	Timed Computational Tree Logic.
TDA	Test-Driven Approach.
TDD	Test-Driven Development.
TDM	Test-Driven Modeling.
TDMS	Test-Driven Modeling Suite.
TD-DSE	Test-Driven Design Space Exploration.
TD-MBSE	Test-Driven, Model-Based Systems Engineering.
TDMBWE	Test-Driven Model-Based Web Engineering.

TDSC	Test-Driven Scenario Modeling.
Tech	Technology, abbreviation for.
Test-driven	Development method where test cases are written before models or code of the SOI.
Test system	System that is used for testing devices or SOIs.
TFS	Microsoft Team Foundation Server.
TLM	Transaction-Level Modeling.
Top-down	Modeling method where modeling starts at the concept level and progresses by adding details at increasingly lower levels, in contrast to bottom-up.
Traceability	The ability to trace artifacts to requirements of originating stakeholders.
UCD	Use Case Diagram.
UI	User Interface.
UID	User Interaction Diagram.
UML	Unified Modeling Language (OMG standard).
UPPAAL	Tool for modeling and verifying formal models, developed jointly at UPPsala University and AALborg university.
UPPAAL-SMC	UPPAAL tool variant for Statistical Model Checking.
UTP	UML Testing Profile (OMG standard).
Validation	Method to determine if requirements, designs or implementations conform to the wishes of stakeholders.
VDM	Vienna Development Method, modeling languages.
Verification	Method to determine if designs or implementations conform to the requirements.
VeriTAS	Modeling tool for the TAS method, see also SimTAny.
VHDL	Very High Speed Integrated Circuit Hardware Description Language
Virtual prototype	Realistic computer model of device or system.
VSL	Value Specification Language (OMG standard).

WebML	Language for modeling web applications.
WM	Wireless Microphones, same as SM.
WoT	Web of Things.
WS	Web Services.
WSM	Weight Scoring Method, in tool selection.
XP	Extreme programming.
xtUML	Executable UML models.

Chapter 1

Introduction

Embedded systems include all types of products that contain a central electronic processor with associated software, except actual computer systems. The processor units of embedded systems are typically used for controlling and monitoring other components of the products and for dedicated computing beyond a simple monitoring-controlling scheme. The spread of embedded systems is growing rapidly at an annual growth rate of approximately 9% from a level of around 160 billion Euro in 2009 [7].

The complexity of embedded systems is also growing rapidly at annual rates of 10 to 30% for software [7], while hardware still follow Moore's Law [8][9], doubling approximately every 18 to 25 months. This shows that software lacks behind hardware with the consequence that the possibilities offered by rapid hardware development cannot be fully exploited at the system level. To utilize hardware better, one must therefore either improve the software development rate or move functionality from software to hardware. In both cases, we need a more holistic system perspective and improved development processes and methods.

Previously, systems were often independent of the surroundings and communicated only with a limited number of systems using fixed connections. Many modern systems, in contrast, are often part of systems-of-systems and exchange information with the surroundings via the Internet and other open communication technologies. Such complex smart systems have the potential to provide features that are more interesting but they require more rigorous development methods to ensure reliability.

Today, embedded and smart systems are often developed using document-based system engineering (DBSE) methods. Requirements, designs, tests, etc., are described using plain text, figures and tables. Common office applications are typically used for creating and maintaining such documents. DBSE is easy to implement, but has numerous drawbacks. The inability to predict the system behavior based on simulations constitute a particular problem, [10].

Many challenges are associated with the development of embedded and smart systems. The number of potential defects and faults is often substantial, which leads to a need for an even greater number of test cases with a complexity and size that often exceeds the system itself [10].

With DBSE, it is not possible to run simulations or execute test cases to find and correct errors at an early stage. By great and thorough effort, however, the industry has repeatedly shown that it is possible to create at least simple systems of good quality.

Engineering of large complex systems requires a different approach to ensure quality and development efficiency. This has led to various agile methods such as extreme programming (XP) and test-driven development (TDD) that have been used with good but varying results [11]. Unfortunately, these methods do not solve the basic problem of predicting the behavior of integrated systems at an early stage before they are implemented and can be subjected to testing. The soundness of requirements and intended system designs can only be demonstrated or proved by simulating models of the overall integrated system.

Dividing the development of constituent technologies (e.g. hardware, software, mechanics, etc.) at an early stage, introduces significant risks into the development projects. Insufficient system performance or even incompatibilities between parts and components may result from separating the engineering disciplines. Such errors are typically found very late in the developments process. This may lead to poor product quality, increased development time, financial losses, etc. Cancellation of projects may be required in severe cases.

Models have been used in various engineering disciplines since antiquity. Good system models make it possible at an early stage to predict system behavior and performance, leading to improved development efficiency and better products. There are many good reasons for and a growing tendency to replace DBSE with model-centric methods such as model-based system engineering (MSBE), model-driven design (MDD) and model-driven engineering (MDE), [12].

Several MBSE methodologies such as the embedded computer-based system analysis method (ECSAM) [10] and the object oriented system engineering method (OOSEM), see [13] or [14], have been proposed and used occasionally, but seems not to be widely used [15]. There are numerous modeling languages available to implement various model-based methods, including: ECSAM [10], SystemC [17], UML [18], xtUML [19], SysML [13], Matlab with Simulink [20] and Stateflow [21] and other more specialized modeling languages [15]. For each of these modeling languages there is at least one available tool [15]. Different combinations of methodologies, modeling languages and associated tools are used for different purposes with distinct characteristics, advantages and disadvantages. However, as individual technologies standing alone they cannot cover all modeling needs for companies such as GN Hearing due to limited expressiveness, unnecessarily complicated syntax and semantics. Currently, there are no straightforward and commonly accepted ways of combining models in different languages into a single overall system model, [15] [16]. Thus, there is a need for investigating how to combine, integrate and use different modeling technologies to enable systems engineers to validate and verify system concepts before commencing development of hardware and software.

The transition from DBSE to MBSE can be rather overwhelming due to the tremendous number of available methodologies, modeling languages and tools. To ease such transitions, this PhD project includes investigations of methods to identify feasible modeling setups based on thorough analyses of the actual modeling needs and of the possibilities offered by different modeling technologies.

From TDD it is known that the test code can be considerably more complex than the code used in the actual product [7]. A similar relationship can be expected for system models and the test cases that must be made to verify the models. Many MBSE methods focus on the system models and ignore the test cases to a large degree. At best, test cases are created afterwards by a systematic analysis of the models [10]. However, experience shows that there are advantages of developing test cases before models, [11]. Several methods for test-driven modeling (TDM) exist, but none have gained popularity or been widely adopted by system engineers for several reasons. The advantages of TDM may not have been demonstrated and communicated effectively or the need may not have emerged yet for the average system engineer. Early adopter may have hesitated because of severe drawbacks such as the need for specialized and complex modeling languages and tool chains. This PhD project includes investigations of new methods for test-driven, model-based systems engineering (TD-MBSE) that with luck will gain more widespread acceptance.

The overall purpose of the PhD project is to investigate methods that facilitate the transition from documents to models. One aim is to develop methods for identifying and implementing suitable modeling setups. Another aim is to create a prototype of a development tool or a modeling framework that supports TD-MBSE at GN Hearing.

The remaining parts of this chapter are structures as follows: Section 1.1 introduces the company GN Hearing and shows how the PhD project fit into the organization.

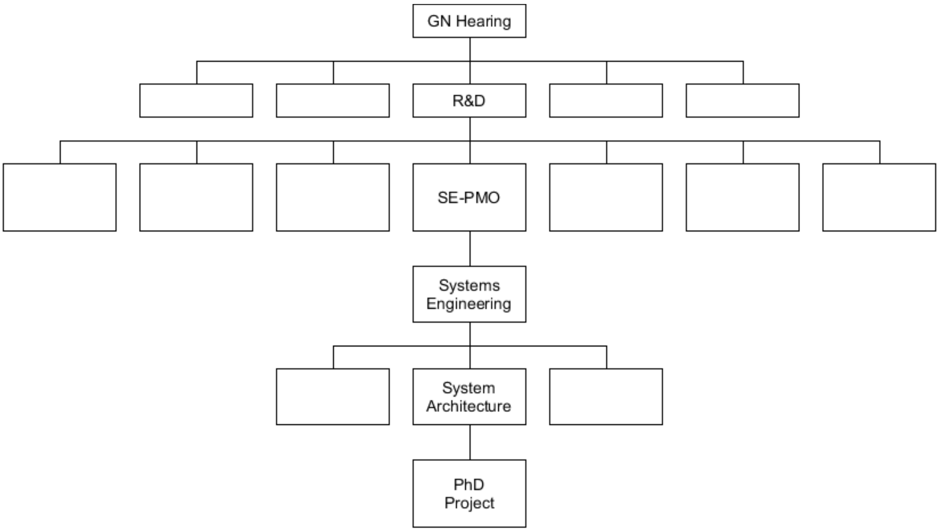


Figure 1.1: GN Hearing organization (simplified, confidential).

Section 1.2 describes the market situation. Section 1.3 describes the strengths, weaknesses, opportunities and threats (SWOT) for the company. Section 1.4 describes the characteristics of the systems of interests. Section 1.5 describes the research questions addressed in the PhD project. Section 1.6 describes the research methods. Section 1.7 describes the hypotheses. Section 1.8 describe the stakeholders involved in the PhD project. Section 1.9 describes the structure and content of the remaining parts of the thesis and section 1.10 finally lists the references cited in this chapter.

1.1 GN Hearing

GN Hearing A/S is part of the company GN Store Nord A/S. The products and services offered by GN Hearing are hearing systems consisting of hearing instruments, remote controllers, streaming devices, auxiliary devices, mobile applications and cloud services, as well as fitting software and associated equipment.

Figure 1.1 shows a simplified organization chart of GH Hearing. The PhD project is associated with the System Architecture team in the Systems Engineering department, which is part of the SE-PMO (systems engineering and project management office) department in R&D (research and development) in GN Hearing. The project is related to other engineering disciplines in the R&D department, e.g. software development and hardware platform development.

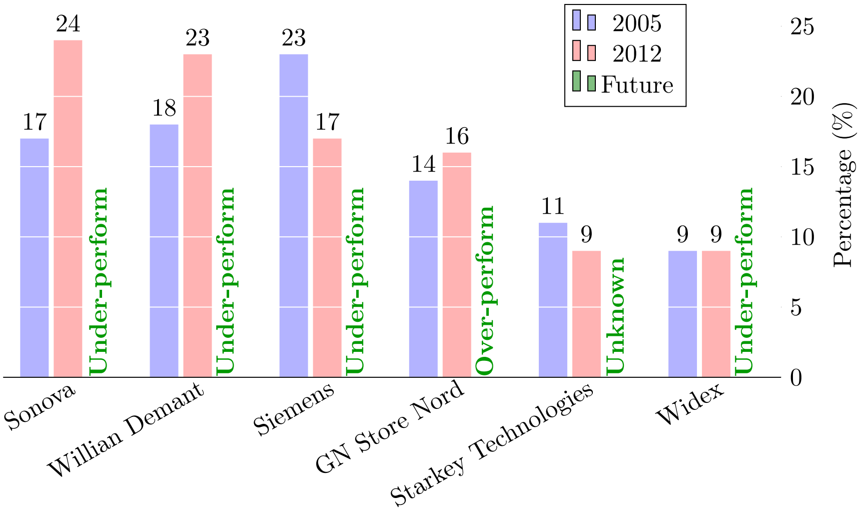


Figure 1.2: Hearing aids market shares. Sonova is the owner of Phonak et al. William Demant is the owner of Oticon et al. GN Store Nord is the owner of GN ReSound (now GH Hearing) et al. Source: Kirkwood [22].

1.2 Market situation

10.8 million hearing aids corresponding to a whole sale value of 5.4 billion USD were sold in 2012 according to [22]. 45% were sold in Europe, 29% in North America and 26% in the rest of the world. The total market is expected to double over a 25-year period corresponding to 3% yearly growth driven by a similar growth in main user group of people over 65 years [23]. Approximately 20% of hearing impaired persons use hearing aids [24]. 98% of the units sold are manufactured by six large companies [22] with market shares as shown in figure 1.2. From the figure, it is seen that GN Hearing was expected to gain market shares in the future[1]. The situation may have changed since 2013 because the competitors of GN Hearing have acquired wireless and other technologies that make their products more attractive.

[1] The analysis ([22]) was published in 2013.

1.3 SWOT analysis

The strengths, weaknesses, opportunities and threats of the company were analyzed at the beginning of this project to outline the current situation of the company. The SWOT is company confidential and cannot be shown in this thesis.

1.4 Hearing systems

The fundamental purpose of hearing systems is to remedy hearing impairment by compensating for the end-users hearing loss. The hearing systems from GN Hearing are enriched with many features that allow end-users to control their hearing instruments remotely and to connect directly to various (electronic) sound sources. Hearing systems also typically include features that allow hearing care professionals (HCP) to calibrate and adjust the hearing aids to the specific needs of the end-users. The ability to connect hearing systems to the Internet allows features that are more advanced, such as remote haring aid firmware updating, remote fine tuning of hearing aids, etc.

Figure 1.3 shows an example of a complex hearing system from GN Hearing. From the figure, it is seen that hearing systems may include many different types of components and means of communication between the parts.

The central devices of hearing systems are the hearing instruments (HI), which provide amplification of sounds, tinnitus management, audio control, and other features. HIs contain advanced electronics and electro-acoustical components to implement these and other features.

Advanced HIs can communicate with other devices and systems such as programming devices (Airlink-2 and Speedlink), mobile devices, remote controls (RC), Bluetooth bridging devices (BTB), wireless microphones (WM) or spouse microphones (SM), stationary audio streaming devices (SAS) that are connected to TV sets, HIFI equipment, computers or similar electrical sound sources. HIs are finally able to receive signals from frequency modulated (FM) systems and magnetic loop systems. The HIs can be connected to the Internet through mobile devices.

The communication between components is facilitate by several types of connections: Bluetooth (standard and Low energy), Proximity (2.4 GHz proprietary standard of GN Hearing), WAN/WLAN, Wired, and magnetic induction.

The components and communication paths of hearing systems from GN Hearing are

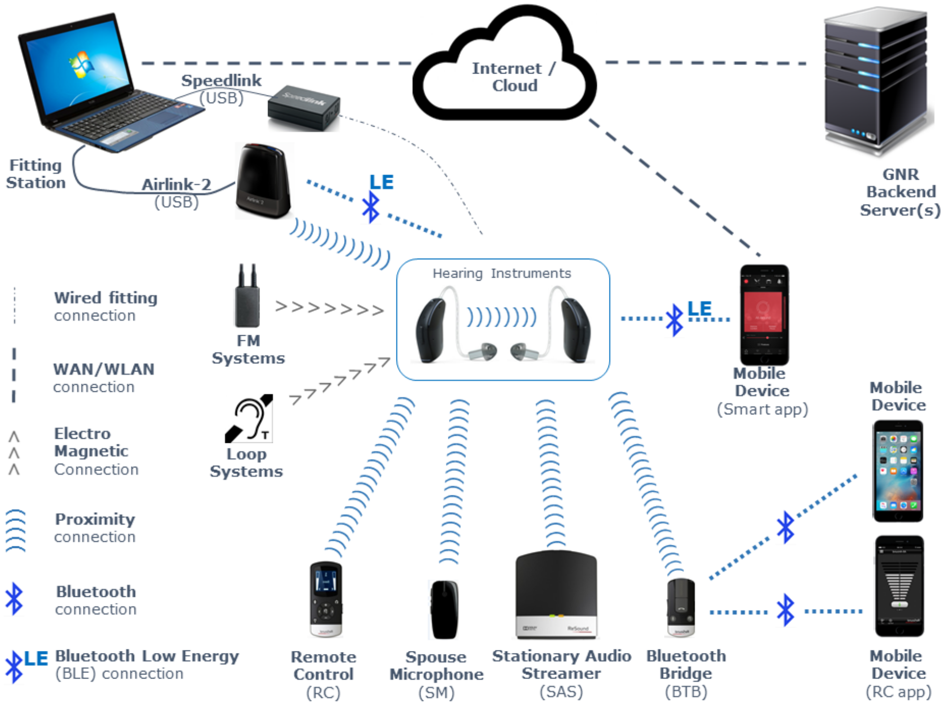


Figure 1.3: Example of hearing systems from GN Hearing.
Source: GN Hearing R&D Systems Engineering.

further described in chapter 2 in [1].

The characteristics of systems determines which technologies can be used for modeling and simulating. Systems classification keywords (section 2.4) can be used to determine the overall characteristics of hearing systems. The descriptions of typical characteristics of embedded systems (section 3.2) and smart systems (section 4.2) are additionally used for the following characterizations.

A hearing system consisting of two hearing instruments, a remote control, an extra external microphone, and a stationary audio streamer can be classified as a system of embedded sub-systems. Concerning hardware, the parts includes analog and digital electronics, batteries, mechanical and electro-acoustical components. Concerning software, the parts contains operating systems, system logic, user interfaces and digital signal processing algorithms. Environmentally, the system includes acoustical and electrical sound sources, as well as human operators.

The HIs, RC, SAS, WM, BTB have characteristics of heterogeneous systems, whereas mobile applications, fitting software and cloud services have characteristics that are closer to homogeneous systems.

Performance is very important especially regarding power consumption and sound quality. The size and form-factor of the HIs are equally or more important for many users. The systems has critical elements regarding safety, security or timing. Concerning safety, the hearing instruments are not allowed to damage the ears of its users by excessively loud sounds. Concerning security, the systems must resist any attempt to interfere with safe operation of the hearing systems. Concerning timing, the devices must conform to the timing constraints of internal electronic component and of the external communication protocols.

The business model includes dispensers (hearing care professionals) to calibrate the hearing aids for the end-user. The dispensers pose special requirements that must be included in system models. The usage of the systems may include scenarios where some parts of the system are discarded or replaced with new parts while other parts are kept.

The hearing system may be part of a larger system where third party gadgets such as smart-phones or tablets are used to mediate web services and cloud solutions. The use of third party devices and services adds the characteristics of systems of system (SoS). Interaction with large crowds of users adds the characteristics of internet of things (IoT), smart systems, and big-data systems.

The modeling complexity must therefore be considered as medium to high. The simulation complexity on the other hand must be consider as high to very high, because it requires several modes of computation (e.g. continuous time and discrete events) to simulate the overall system. However, in many cases, it will be possible to abstract and simplify models such that the systems can be expressed using only the discrete event formalism.

The selection of suitable technologies and methodologies for modeling of such systems is discussed in chapter 2.

1.5 Research questions

The overall purpose of the PhD project is to investigate modeling methods, languages, tools and processes to facilitate the transition from documents to models. This thesis addresses two major research questions:

RQ 1 How can organizations select and implement tools for (model-based) systems engineering?

RQ 2 How and to what effect can test-driven methods be applied to model-based systems engineering?

The first question, RQ 1, is relevant for all organizations that embark on the transition from documents to models. RQ1 can be elaborated into the following set of research questions concerning selection of MBSE tools:

RQ 1.1 Which methods have previously been applied to selecting, implementing and using systems engineering tools?

RQ 1.2 Which limitations or disadvantages of existing methods for selecting and implementing systems engineering tools necessitate modified or new methods?

RQ 1.3 How can tool solutions be verified and validated before they are implemented and used on a daily basis?

RQ 1.4 What can be achieved by using existing or newly proposed methods for selecting and implementing tools?

RQ 1.5 What are the advantages of the proposed method for selecting, implementing and using tool solutions?

RQ 1.6 What are the disadvantages and limitations of the proposed method for selecting, implementing and using tool solutions.

The second question, RQ 2, is relevant for organizations that require models of high quality and trustworthiness. RQ 2 can be elaborated into the following set of sub research questions concerning test-driven modeling of embedded and smart systems:

RQ 2.1 How have test-driven methods previously been applied to systems modeling?

RQ 2.2 Which limitations of existing methods for test-driven systems modeling necessitate modified or new methods?

RQ 2.3 What are the advantages, disadvantages and limitations of different verification techniques for test-driven systems modeling?

RQ 2.4 What can be achieved and what are the experiences of using test-driven modeling for design of embedded and smart systems?

RQ 2.5 What are the advantages of the proposed methods for test-driven systems modeling?

RQ 2.6 What are the disadvantages and limitations of the proposed methods for test-driven systems modeling?

Questions regarding the effectiveness, advantages and disadvantages of using DBSE versus MBSE versus TD-MBSE have deliberately been avoided because such question are very difficult to answer in our case due to the following reasons:

1. We do not have a solid base line to compare with.
2. We will not be able to conduct enough modeling cases to get results with sufficient statistical significance.
3. We do not have the resources to conduct comparative studies.
4. It is very difficult to define metrics and measure the causes of success due to the uniqueness of each case as described by Muller [26, p. 1100]: "It is close to impossible to make hard claims of success related to specific methods, since we cannot unravel the causes of success."

1.6 Research methods

The research methods used in this study are devised to give adequate answers to the research questions taking the nature and characteristics of the project and its unavoidable constraints into consideration.

This research project has characteristics of or relates to numerous sciences including:

- Natural science:
  - Human biology and audiology.
- Social science:
  - Communication, psychology and sociology.
  - Economics, business and industrial sciences.
- Formal science:

  - Systems theory and science.
  - Computer science.
  - Mathematics, logic and statistics.

- Applied science:
  - Systems engineering.
  - Systems modeling and simulation.
  - Formal and statistical model checking.
  - Hardware engineering.
  - Software engineering.

The systems under consideration for modeling at GN Hearing are fundamentally of a technological kind as described in section 1.4. However, the characterization of hearing impairment and interactions between systems and their environments relate more to natural science. How users interact with the systems under considerations also has elements that are best described by social and applied science.

Systems engineering and MBSE have strong elements of the formal and applied sciences. However, systems engineers are heavily involved in communication with stakeholders, driving processes, etc. Such skills are best described by the social sciences.

Modeling languages, tools, methodologies and processes are the cornerstones of MBSE. These elements fall naturally into the group relating to formal and applied sciences. However, their conception are always a product of the fusion of ideas and opinions of various stakeholders. Therefore, their characteristics also relate to social science.

The choice of research methods was restricted by the following constraints: Minimal involvement of employees at GN Hearing to avoid reduced productivity of regular R&D activities. The employees can therefore only be involved as sources of information or as evaluators of results. All activities of investigating, modeling, simulating, etc., must be conducted by the PhD student (me). Limited resources and time therefore prevent certain research methods such as comparative studies.

The characteristics and constraints of the project as described above dictates the use and fusion of various methods. The research method used in this study are:

1. Literature studies of DBSE, MBSE, tool selection, modeling, simulating, verifying, model checking, test-driven methods, etc.

2. Investigations of modeling disciplines that relate to different aspects of MBSE for the purpose of identifying the needs and requirements to feasible MBSE setups (modeling languages, tools, methods, and processes). These investigations are mainly based on literature studies.

3. Investigations of available modeling technologies (modeling languages, tools, methods and processes) for the purpose of being able to identify the best candidates for the relevant MBSE scenarios. These investigations are based on literature studies and modeling experiments to evaluate the technologies. These investigations are the foundation upon which the proposed methods are build.

4. Propose and test methods for selecting and implementing modeling setups. The proposals are mainly based on literature studies. Testing and evaluating the proposed methods are based on case studies conducted at GN Hearing. One case study concerns selecting a traceability tool used throughout R&D. Another case study concerns selecting modeling tools for systems engineering.

5. Propose and test methods for TD-MBSE. The proposals are based on the findings from literature studies and investigations of modeling disciplines, technologies, methods and processes. Testing and evaluating the proposed methods are based on case studies conducted at GN Hearing. One case study concerns test-driven modeling and design space exploration of an embedded system of medium complexity. Another case study concern a smart system. The size and complexity of this system challenges any attempt to obtain trustworthy predictions about the functionality and performance of the system.

6. Identify and analyze expected as well as unexpected findings from the case studies.

7. Report research results as papers, thesis chapters and company presentations.

Other research methods might also have been relevant, such as:

- Observational studies of systems engineers during modeling sessions to gain more insight into the effects of using the proposed methods.

- Surveys for obtaining data from systems engineering stakeholder to gain insight into the effect of using the propose methods.

- Triangulating data obtained from different case studies and other sources to compare the results of using different methods, e.g. DBSE versus MBSE versus TD-MBSE, or different forms of TD-MBSE.

- Series of case studies to explore the proposed methods from different angles and for obtaining statistical significant results.

- Longitudinal studies to gain insight into the long-term effects of using the proposed methods, e.g. changes in productivity, work satisfaction, etc.
- Action research with the aim of changing the organization or work methods or processes. The case studies include some elements of action research because they aim to pave the way for changes but do not include all elements such as client involvement.

However, these and other research methods [27, chp.5] were deliberately avoided for several reasons. They required more resources and involvement from GN Hearing than possible. They concern different research questions than those that were selected for the project. The 7 selected research methods are believed to provide satisfying answers to the research questions described in section 1.5.

1.7 Hypotheses

The following research hypotheses will be assumed concerning RQ 1:

RH 1.1 Several methods for selecting tools that relate to some parts of systems engineering exist but are inadequate for general selection of MBSE tools.

RH 1.2 New methods are needed for systematic analyzing tool requirements to develop relevant selection criteria for MBSE tool selection.

New methods are needed for evaluating, selecting, and implementing tool solutions to accommodate situations not previously addressed.

RH 1.3 Correlating requirements and tool capabilities to identify non-conformance to different groups of requirements can be used for avoiding tool solutions that cannot be verified.

Tailoring the elicitation of tool requirements to the individual organization increases the likelihood of obtaining solutions that can be validated.

RH 1.4 Using systematic method for selecting and implementing tools increases the likelihood of obtaining good solutions that are accepted by the stakeholders.

RH 1.5 The research will provide comprehensive methods that allow successful selection, implementation and subsequent use of MBSE tool solutions.

RH 1.6 The disadvantages of the proposed methods will not cause the organizations to select and implement infeasible tool solutions.

The following research hypotheses will be assumed concerning RQ 2:

RH 2.1 Several test-driven modeling methods exist but they are inadequate for modeling and verifying hearing systems.

RH 2.2 New methods for test-driven modeling and verification are needed for allowing system engineers to guarantee system functionality and performance of embedded systems.

New methods are needed for modeling, simulating and predicting functionality and performance of large and complex smart systems.

RH 2.3 Formal and statistical model checking can be used in methods for test-driven modeling and verification of functionality and performance of medium complex embedded systems.

Simulating can be used in methods for test-driven modeling and verification of functionality and performance of large complex smart systems.

Some systems are too large and complex to be verified with model checking or simulation. The functionality and performance of such systems can be predicted by mathematical forecasting based on simulation results for smaller and simpler versions of the system of interest.

RH 2.4 The use of test-driven modeling will increase productivity, improve quality and reduce the cognitive load during modeling.

The modelers will gain deeper insight into the systems of interest and they will be able to answer questions that otherwise cannot be answered sufficiently.

The modelers will be able to explore the design space and evaluate different versions of the intended systems before starting the actual development of system parts.

RH 2.5 The research will produce test-driven methods that has several advantages compared to previous development methods (including previous test-driven methods).

Combining different verification methods will allow modeling of a wide variety of systems and provide more trustworthy verification results.

RH 2.6 The disadvantages of the proposed test-driven methods will not negatively affect creativity, productivity, reliability, or trustworthiness of the verification results. Disadvantages will only concern verification time and required computer resources.

Notice that each research question has exactly one research hypothesis (e.g. RQ 1.3 <—> RH 1.3).

1.8 Stakeholders

Several stakeholder are involved in or affected by the project. The roles and value propositions for the various stakeholders are described in the following subsections.

1.8.1 PhD student

Role: My role in the PhD project is to conduct research within the discipline of model-based systems engineering and disseminate the result to the company, the university and other involved parties. For the university, I shall build knowledge about MBSE and disseminate in the form of published articles. For the company, I shall conduct experiments that relate to the core business of the company and disseminate in the form of information meetings, workshops and training. For the ITOS and Infinit projects (see section 1.8.5), I shall provide content for the Infinit EmbeddedWiki [28] and the ITOS Fieldbook [29] in collaboration with other industrial projects.

Value proposition: The PhD project provides me a unique opportunity to pursue and explore the field of model-based systems engineering and obtain a PhD degree for improved future career prospects.

1.8.2 GN Hearing

Role: The project is initiated by the company GN Hearing A/S (GNH). GNH is a manufacturer of hearing systems including hearing aids, accessories and auxiliary systems. The complexity of the hearing systems is continually growing. GNH has therefore seen a need for a more formalized systems engineering methodology that allows them to develop future systems. R&D Senior Vice President Lars Lindqvist from GNH is the company supervisor on the project and Systems Engineering Manager Flemming Schmidt from GNH is the company co-supervisor. The role of GNH is to provide the cases and resources that are needed to conduct the research experiments and case studies.

Value proposition: The PhD project provides the company an opportunity to investigate how to introduce modeling into systems engineering at a low level of costs and risks. The project will include experiments that can be utilized as demonstrator to reduce potential resistance towards the introduction of new methods, processes and tools. The company will be able to keep their technological competitiveness by building skills in test-driven, model-based approaches to create better and more attractive product offerings faster, cheaper and with lower levels of risks. The company will be able to offer jobs that are more attractive when document authoring is replaced by modeling. The company will finally be able to gain a reputation for using the best methods to design their systems.

1.8.3 Employees

Role: Several employees at GNH were involved in this PhD project. From the organization chart in figure 1.1, it is seen that PhD project is directly associated with the Systems Architecture team in the Systems Engineering group. This team is highly involved in the project. The project also affects and partly involves the Requirements Engineering and System Verification teams. Employees from various teams in the Software Development department are also highly involved. Employees from other departments (particularly Hardware Platform Development) may ultimately be affected by the methods gain by the project.

Value proposition: The systems engineers and developers will increase their skill sets in a field of increasing importance (model-based systems engineering), which in turn will increase their attractiveness and career prospects. The job content will be more interesting for engineers and similar personalities, when documents are replaced by models. Using the best methods allows employees to perform at their best and to be perceived as professionals. Collaborating on models finally allow social integration of employees from different teams.

1.8.4 Technical University of Denmark

Role: The project is conducted in collaboration with the Technical University of Denmark (DTU) at the Department of Applied Mathematics and Computer Science (DTU-Compute) in the Embedded Systems Engineering (ESE) section. Professor Jan Madsen at DTU-Compute is university supervisor on the project and Professor Paul Pop is the university co-supervisor. The role of the university is to provide access to state of the art knowledge within the area of systems engineering. The university will also provide or facilitate the teaching and training needed to undertake the PhD project successfully. Finally, the university grants the PhD degree.

Value proposition: The PhD project provides ESE with opportunities to build competences in the field of general and test-driven model-based systems engineering to complement their expertise in embedded systems design. This new knowledge can be commercialized as new courses or educations in model-based systems engineering.

1.8.5 DI ITEK ITOS & Infinit

Role: The project is co-funded by the Confederation of Danish Industry (DI) at department for IT, Tele, Electronics and Communications (ITEK) under the Industrial technology and Software (ITOS) project. The purpose of the ITOS project is to create knowledge and enhance skills and techniques used to develop embedded technologies (systems with embedded computers). Senior Consultant Henrik Valentin Jensen from DI is the project manager for ITOS project and associate Professor Ulrik Nyman from Aalborg University (AAU) is the coordinator of ITOS activities. The ITOS project is associated to numerous organizations and projects regarding model-based systems engineering including: (1) PhD project at Technical University of Denmark (DTU) in collaboration with MAN Diesel conducted by PhD student Nicolai Pedersen. (2) PhD project at Aalborg University (AAU) in collaboration with Seluxit conducted by PhD student Thomas Pedersen. (3) PostDoc project at Aarhus University (AUC) in collaboration with Terma conducted by PostDoc Sune Wolff. (4) EmbeddedWiki project funded by Infinit and managed by Project Manager Mads Kronborg Ågesen. Other stakeholders were involved in the ITOS and Infinit projects, e.g. as supervisors. Collaboration between all involved parties produced the EmbeddedWiki [28], ITOS Fieldbook [29], ITOS Workbook [30], and other ITOS publications.

Value proposition: The PhD project provides the ITOS and Inifinit projects a very general knowledge about model-based systems engineering. This PhD project is much more general than the other industrial cases in the ITOS project and will therefore be of more interest to many users of the ITOS publications and the Infinit EmbeddedWiki. This PhD project provides general knowledge about model-based systems engineering to the other industrial cases, which are more focused on specialized modeling methods, tools and application domains.

1.8.6 Tool providers

Role: Tool providers and open source tool projects are not directly involved in the project, but may be affected by the methods gained from the project.

Value proposition: The proposed method for test-driven modeling may be leveraged by tool providers to develop plugins or extensions to existing modeling tools for the benefit of their customers.

1.8.7 INCOSE

Role: The Danish chapter of the International Council on Systems Engineering (INCOSE) is an important stakeholder that may be affected by the project even though they are not directly participating in the project. Learning model-based systems engineering methods is difficult and the number of skilled people is probably too low to satisfy the actual need both in Denmark and internationally. INCOSE provides a great network for promoting model-based systems engineering.

Value proposition: The method for selecting, implementing and using model-based systems engineering tools may be leveraged by INCOSE to help organizations to commence the transition from documents to models.

1.8.8 Embedded systems industry

Role: The embedded systems industry is not directly participating in the project but the research relate strongly to objectives of the industry, which may therefore be affected by the results of the research.

Value proposition: The PhD project provides the embedded systems industry methods that can be applied to many different domains. The knowledge produced by the project will allow the industry to efficiently create systems that are larger, more complex, and of higher quality. The required skill level is lowered when methods that are more systematic are adopted such that more people are able to contribute to successful systems engineering. Using the best methods will also be attractive to students, which will contribute to preventing the lack of skilled people in the future.

1.8.9 Society

Role: The project is co-funded by the Innovation Fund Denmark. The society therefore provide means for conducing this PhD project but is not otherwise involved in the research. However, society as such may be affected by the results of the project.

Value proposition: The PhD provides the society a knowledge base that will enable it to develop large complicated system for the benefit of all citizens. It will contribute to promoting useful educations, interesting jobs and meaningful lives. Increased tax revenue from a successful embedded systems industry is will also benefit society.

1.9 Thesis

1.9.1 Purpose and scope

The purpose of the thesis is to document the research undertaken during and relating to the PhD project " Test-Driven, Model-Based Systems Engineering ". A vast amount of work relating to the purpose of the project (e.g. learning the subtleties of numerous modeling methods and technologies) has been undertaken as preparation for the execution of the case studies and other research activities. Some of this preparatory work is has produced results that are included in the chapters and appendices of this thesis. However, most parts of the preparatory work has not produced new results of relevance for stakeholders or academia, so these parts are not included in the thesis. The scope of the thesis is limited to include descriptions of the research that was conducted to provide answers for the research questions and evaluations for the hypotheses.

This PhD project was initiated by GN Hearing with the idea of addressing some challenges concerning development of future embedded and smart systems. The overall purpose of the project is therefore to conduct research that allow GN Hearing to identify potential solutions to these issues.

The scope of the project is limited to the disciplines of model-based systems engineering where major improvements are considered reachable. Specialty engineering disciplines such as hardware or software development have occasionally been involved in the project. However, the project has a strong focus on methods that allow systems engineering to commence the transition from documents to models.

The scope of project is also limited to systems of interest for GN Hearing, i.e. products and services involved in complex hearing systems. Large and complex hearing systems include elements such as crowds of users, devices, clouds, web services, etc. This project has a focus on embedded and smart systems, because most devices are in the embedded systems category and because hearing systems currently and onwards are evolving from simple predictable systems into large and complex smart systems. Other aspect of hearing systems are managed within specialty engineering and are therefore excluded from the project.

The PhD project aims at (1) developing methods that allow GN Hearing and similar companies to commence the transition from documents to models, and (2) developing a test-driven, model-based systems engineering framework and accompanying development tool set for creating, verifying and validating system models of high quality. To obtain viable long-term industrial solutions, it is preferred to use and tailor proven tools instead of developing specialized tools that may be hard to maintain. The focus is thus centered on methodologies and processes rather than tools and other forms of supporting technologies.

1.9.2 Style and conventions

This thesis conforms to the traditions at the Embedded Systems Engineering section at DTU Compute. The term "we" and "us" are used instead of "I" or "me" to implicit acknowledge that the work presented here builds upon the work of others. However, all research described in this thesis is conducted by the author (me), albeit under the guidance of the supervisors from DTU and GN Hearing.

1.9.3 Abbreviations and terms

Abbreviations are defined directly in the text whenever needed. The full term is written first followed by the abbreviation in parentheses. Previously defined abbreviations may be repeated or redefined whenever convenient. The abbreviations and terms are furthermore listed in the vocabulary starting on page xxi.

1.9.4 Thesis outline

Chapters

Chapter 1 introduces the PhD project. It briefly describes the company GN Hearing and the market situation. It further describes research questions, methods and hypotheses. The roles of and value proposition for the stakeholders of the project are described. The content and structure of the thesis is finally outlined.

Chapter 2 describes the research concerning methods for selecting, implementing and using MBSE tools. Review of related work is described and new methods are proposed, tested and evaluated by case studies. The chapter includes classifications of systems and modeling disciplines as prerequisites for using the proposed methods. Results, observations, learning points, advantages, disadvantages, limitations, potential improvements, and suggested future work are discussed and concluded.

Chapter 3 describes the research concerning test-driven modeling of embedded systems. Prior work relating to test-driven modeling is reviewed. New methods for test-driven modeling and design space exploration are proposed, tested and evaluated by a case study. Results, observations, learning points, advantages, disadvantages, limitations, potential improvements, and suggested future work are discussed and concluded.

Chapter 4 describes the research concerning test-driven modeling of smart systems. Prior work relating to methods and technologies for modeling and simulating systems that resembles smart systems is reviewed. New methods that include formal model checking, simulating and mathematical forecasting are proposed, tested and evaluated by a case study. Results, observations, learning points, advantages, disadvantages, limitations, potential improvements, and suggested future work are discussed and concluded.

Chapter 5 discusses and summarizes the research results. The chapter includes guidelines for implementing selected MBSE tools, using various modeling disciplines, test-driven modeling of embedded and smart systems. The guidelines supplement the proposed methods by giving advice on when, for what and by whom the methods be applied. Answers to the research questions are provided in the results section. Observations, problems, open issues, future work, and potential impact of the project are summarized and discussed. Evaluations of the research hypotheses are included in the final conclusion.

Cited references are available at the end of each chapter.

Navigation

A summary of this thesis is available on page i, a summary in Danish is available on page iii, and the preface is available on page v.

The papers that are included in this thesis are listed on page vii.

Acknowledgments of important stakeholders are available on page ix.

Lists of contents, figures and tables are starting on page xi, xvii and xix, respectively.

A vocabulary of abbreviations and major terms used in this thesis is started on page xxi.

Technical reports

The content presented in this thesis is further elaborated in a series of associated technical reports as described below.

Chapter 2 in [1] describes the product parts of hearing systems from GN Hearing.

Chapter 2 in [2] contains guides and detailed descriptions of how to conduct model-based systems engineering based on experience obtained from experiments that were conducted during the project.

Chapter 2 in [3] contains the details of the analysis that was conducted to identify a set of feasible modeling technologies (used in chapter 2).

Chapter 2 in [4] contains model diagrams, source code listings, analysis queries, and verification results for various experiments and the case discussed in chapter 3.

Chapter 2 in [5] contains details concerning verification methods for smart systems and suggested improvements of the implementation of the method proposed in chapter 4.

Chapter 3 through 5 in [5] and chapter 2 through 4 in [6] contains model diagrams, source code listings, analysis queries, and verification results for various experiments and the case discussed in chapter 4.

Chapter 3 in [2] contains guidelines for the use of the methods that are proposed in this thesis.

1.10 References

[1] Allan Munck
"Hearing systems."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[2] Allan Munck
"Model-Based Systems Engineering Guidelines."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[3] Allan Munck
"Modeling tool selection."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[4] Allan Munck
"Embedded systems models."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[5] Allan Munck
"Smart systems modeling."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[6] Allan Munck
"Smart systems simulation code and results."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[7] Christof Ebert and Casper Jones.
"Embedded software: Facts, figures and future."
IEEE Computer Magazine, 2009, Volume 42, Issue 4.

[8] Gordon E. Moore.
"Progress in digital integrated electronics."
Electron Devices Meeting. Vol. 21. 1975.

[9] Robert R. Schaller.
"Moore's law: past, present and future."
Spectrum, IEEE 34.6 (1997): 52-59.

[10] Jonah Z. Lavi and Joseph Kudish.
"Systems Modeling and Requirements Specication Using ECSAM: An Analysis Method for Embedded and Computer-Based Systems."
Dorset House Publishing, New York, 2005.
ISBN-10: 0-932633-45-5, ISBN-13: 978-0932633453

[11] Piet Cordemans, Sille Van Landschoot, Jeroen Boydens, Eric Steegmans and Janusz Kacprzyk.
"Studies in Computational Intelligence - Embedded and Real Time System Development: A Software Engineering Perspective: "Concepts, Methods and Principles" - Test-Driven Development as a reliable embedded software engineering practice." (book chapter).
Springer Verlag, 2013, ISSN: 1860-949X.

[12] John Hutchinson, Jon Whittle, Mark Rouncefield and Steinar Kristoffersen.
"Emperical Assessment of MDE in Industry."
ICSE '11, Proceedings of the 33rd International Conference on Software Engineering, 2011.

[13] Sanford Friedenthal, Alan More and Rick Steiner.
"A Practical Guide to SysML: The Systems Modeling Language."
Morgan Kaufmann; 2nd Revised edition (26 Nov 2011), ISBN-10: 0123852064.

[14] Jeffrey A. Estefan.
"Survey of Model-Based Systems Engineering (MBSE) Methodologies."
Rev. B, INCOSE Technical Publication, Document No.: INCOSE-TD-2007-003-01, International Council on Systems Engineering, San Diego, CA, June 10, 2008.

[15] Allan Munck.
"Investigation of formal methods for the planning, analysis, design and verification of complex embedded computer systems."
Master thesis, Technical University of Denmark, Informatics and Mathematical Modeling, 2012.

[16] Albert Albers and Christian Zingel.
"Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML."
Smart Product Engineering - Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11th - 13th, 2013, pp 83-92. Springer-Verlag Berlin Heidelberg, 2013, ISBN: 978-3-642-30816-1

[17] IEEE Standard 1666-2011.
"IEEE Standard for Standard SystemC ® Language Reference Manual."
ISBN 978-0-7381-6801-2 STD97162.

[18] Object Management Group (OMG).
"Unified Modeling Language (UML), Version 2.4.1."
`http://www.omg.org/spec/UML/2.4.1/`

[19] Mentor Graphics: Bill Chown and Dean McArthur.
"The style guide to xtUML modeling", Revision 1.1, November 10, 2011.
"The xtUML modeling guide", Revision 1.1, November 10, 2011.

[20] Mathworks Matlab/Simulink.
"Simulation and model-based design for dynamic and embedded systems."
Website (last visited 2016.01.24):
`http://www.mathworks.se/products/simulink/`

[21] Mathworks Matlab/Stateflow.
"Design environment for developing state charts and flow graphs tightly integrated with MATLAB and Simulink."

Website (last visited 2016.01.24):
http://www.mathworks.se/products/stateflow/

[22] David Kirkwood citing Sanford C. Bernstein.
"Research firm analyzes market share, retail activity, and prospects of major hearing aid manufacturers."
http://hearinghealthmatters.org/hearingnewswatch/2013/research-firm-analyzes-market-share-retail-stores-prospects-of-major-hearing-aid-makers/

[23] Søren Nielsen.
"Dynamics of the hearing aid market."
http://files.shareholder.com/downloads/ABEA-4C7PH1/1474586245x0x392115/e0be1288-fbc0-4ece-aa15-96d5a0432ec5/CMD3_1.pdf

[24] Niels Jacobsen and Søren Nielsen.
"Trends and directions in the hearing healthcare market."
http://files.shareholder.com/downloads/ABEA-4C7PH1/2626050094x0x671887/cb72e485-ebc6-4c72-b4c7-428388a8ee13/CMD_2013_Trends_and_directions_in_the_hearing_healthcare_market.pdf

[25] Nordic Semiconductor.
"Hearing system."
Web resource (last visited 2017.01.26):
http://www.nordicsemi.com/var/ezwebin_site/storage/images/news/press-releases/product-related-news/nordic-2.4ghz-technology-enables-miniature-hearing-aid-to-stream-audio-direct-from-tvs-and-other-consumer-devices/151883-1-eng-GB/Nordic-2.4GHz-technology-enables-miniature-hearing-aid-to-stream-audio-direct-from-TVs-and-other-consumer-devices.jpg

[26] Gerrit Muller.
"Systems Engineering Research Methods."
Conference on Systems Engineering Research (CSER'13), March 19-22, 2013, Procedia Computer Science 16 (2013) 1092 – 1101.

[27] Jill Collis and Roger Hussey.
"Business research: A practical guide for undergraduate and postgraduate students."
Third edition, Palgrave macmillan, 2009.

[28] Infinit.
"EmbeddedWiki."
http://embeddedwiki.cs.aau.dk/wiki/Main_Page

[29] DI ITEK ITOS. *"Fieldbook: Developing embedded systems & smart products in practice."*
English version: `http://publikationer.di.dk/dikataloger/676/`
Danish version: `http://publikationer.di.dk/dikataloger/675/`

[30] DI ITEK ITOS.
"Workbook: Smarte produkter i praksis - worhshop øvelser."
Danish version: `http://publikationer.di.dk/dikataloger/674/`

Chapter 2

Modeling technologies

The transition from document-based to model-based systems engineering is a mandatory exercise for many organizations due to increasing system complexity and escalating requirements regarding increased quality, reduced development time and costs, etc. This transition can be overwhelming due to the tremendous number of methodologies, modeling languages and tools available. Instead of conducting a thorough analysis, many organizations tend to choose a setup that only supports a small part of the immediate needs. However, choosing a wrong setup may have severe impact on the future success of the organization.

This chapter presents a systematic and practical method for selecting systems engineering tools with a focus on modeling technologies (methodologies, languages, and tools). The method describes how to justify and initiate the change project, how to develop tool specifications and evaluation criteria, how to investigate tools and select the best candidates, and how to implement, maintain and evolve the chosen solution.

The entire model-based systems engineering discipline is categorized into distinct subdisciplines for which desired features can be formulated. This grouping allow us to obtain a comprehensive set of requirements that are evaluated for all tool candidates. The evaluation process therefore becomes highly systematic.

Correlating requirements and tool capabilities enables us to identify the best tool for single-tool scenarios or the best set of tools for multi-tool scenarios. For single-tool scenarios, simple ranking techniques are used to identify the best candidates. Analyzing the gaps between required and obtained properties prevents selecting infeasible tools. For multi-tool scenarios, set theory and simple reduction techniques are used to identify the best tool set candidates, and gap analyses prevents selecting infeasible sets of tools.

The method was utilized to select a set of tools that we used on pilot cases at GN Hearing for modeling, simulating and formally verifying embedded systems and complex smart systems.

The remaining parts of this chapter are divided into the following sections. Section 2.1 introduces the background for and the topic of selecting modeling technologies. The related works on tool selection are described in section 2.2. Section 2.3 describes the proposed method for selecting modeling tools. Proper use of the proposed method requires understanding systems under consideration as described in section 2.4 and knowledge of the applicable modeling disciplines as described in section 2.5. Section 2.6 describes the application of the method to selecting a traceability tool for GN Hearing and section 2.7 describes the application of the method to selecting tools for modeling pilot-cases at GN Hearing. Section 2.8 discusses the results of using the proposed method, observations, learning points, advantages, disadvantages, limitations, potential improvements and future work. The section is ended by an overall summary of and conclusion on the chapter. References cited in the chapter are finally listed in section 2.9. This chapter extends and refines the paper [73].

2.1 Introduction

Many companies in the embedded systems sector have dedicated systems engineering (SE) departments to perform the formalized SE activities. Typically, various other employees are also involved in the SE activities of complex systems over the entire life cycle from ideation to disposal. All successful companies involved in embedded systems are performing SE activities to some degree even if they do not have dedicated SE departments.

SE covers a large span of activities. Forty sections are dedicated to describing SE processes and specialty engineering activities in the INCOSE SE handbook [5]. Most engineering activities are transforming artifacts from input sources to output recipients. Typical examples of artifacts include sketches, documents, spreadsheets, presentations, models, mock-ups, virtual and actual prototypes, test systems, and end-products. The quality of the end-products are to a large degree determined by the early SE artifacts.

Historically[1], SE has been managed with simple tools such as pen and paper, typewriters, pocket calculators. Such simple tools have mostly been replaced by computers today. Tools to edit spreadsheets and text documents with embedded images are popular in SE. Some organizations have progressed further and are using tools dedicated to modeling, simulating, and managing of requirements, tests, defects, etc. However, many continue with simpler or fewer tools.

The growing complexity of systems makes it increasingly difficult to conduct successful SE without using proper tools. Requirements management (RM) tools can facilitate structured processes for defining constraints and requirements. Test management (TM) tools can have the same effect on test case definition, test planning and executing. Application and product lifecycle management (ALM, PLM) tools can facilitate integration of several engineering disciplines. Modeling tools can facilitate better communication between stakeholders and support thorough analyses of both problem and solution spaces. Simulation tools can facilitate early identification of problems so they can be corrected before it becomes too expensive or difficult to do so. Sometimes developers ignore requirements and implement what they think is correct. SE tools with traceability features can be used to remedy this problem and ensure that requirements are respected and applied.

Good tools therefore make it possible for organizations to manage processes more efficiently. A wide variety of tools is available for various aspects of SE. However, the sheer number of available tools makes the process of selecting the best tools somewhat intimidating.

Some organizations have fallen into the trap of selecting tools that do not meet all immediate and future needs. Thorough analyses are ignored in such cases and tools seem to be chosen on basis of fragmentary experiences of a few randomly selected people. For modeling and simulation tools, there has also been limited knowledge about available methodologies, modeling languages, supporting tools, required training, costs, risks, etc. This has led to the choice of modeling setups that may not be optimal for current or future modeling needs.

However, selecting the wrong tools may have severe consequences for the success of the companies. Considerable costs may be wasted on licenses, consultants, training, installation, and configurations of tools that do not meet the needs, requirements and constraints of the company. Selecting even the most optimal tools is not sufficient to ensure success, if they are implemented or used inappropriately.

[1] From INCOSE's definition of SE, it can be concluded that many parts of SE have been applied for centuries to analyze, design and realize whole systems even though the modern form of SE originates from the 1930's [5, pp.7-8].

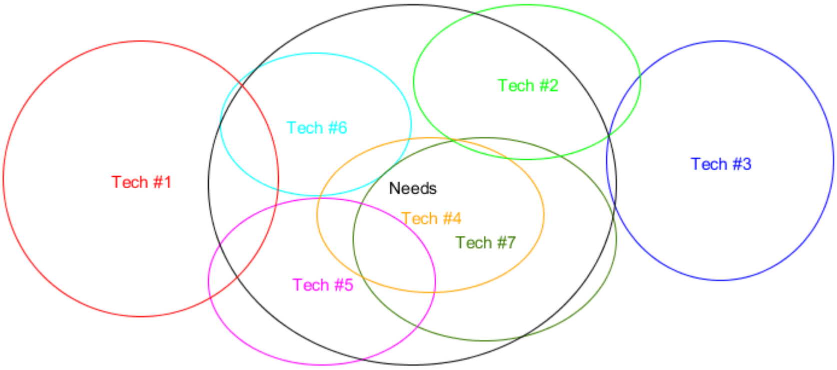


Figure 2.1: Modeling needs and technologies. The black ellipse illustrates the modeling needs. The colored circles illustrate the needs covered by different modeling technologies. None of the technologies covers all needs. Most but not all needs can be covered. The combination of `Tech #2`, `Tech #5`, `Tech #6` and `Tech #7` could provide a satisfying solution. `Tech #1` and `Tech #3` have other capabilities than needed and they are therefore irrelevant. `Tech #4` is redundant in this case.

Finding a feasible setup is basically a matter of identifying a set of modeling technologies that covers the needs as illustrated in figure 2.1, where seven different technologies cover most but not all needs. A valid method must therefore provide means for identifying the needs, identifying the available technologies, and selecting the best alternatives. To ensure successful adoption and use of the selected tools, such a method must also consider how to implement, maintain and evolve the solution. The method presented in this chapter includes all these elements. It was used to select a traceability tool (section 2.6) and a set of modeling tools (section 2.3) for GN Hearing.

The proposed method may be adopted by organizations to select various forms of SE tools including modeling tools. The amount of work that is required to conduct such an analysis should not be underestimated, however. Companies involved in the development of embedded products may be able to re-use the results to select modeling tools, without much further analysis. Other organizations may use the presented results as a starting point, though it may be necessary to verify that the results hold for their cases. Most likely it will be necessary to extend the analyses with their specific needs, constraints and conditions.

2.2 Related work

Searching the literature concerning selection of SE and MBSE tools reveals very little. Tools for various aspects of SE are sporadically mentioned in the INCOSE systems engineering handbook (2011) [5], but selection criteria and methods are not provided. However, the search has identified several articles and book chapters on related issues, which allow the review of existing work regarding:

- Application domains (selection of vendors, simulation tools, requirements management tools, traceability tools, modeling tools, software packages, etc.).
- Evaluation criteria for various domains.
- Selection and implementation methods.
- Specific tool evaluations.
- Lists of tools for selected domains.

Section 2.2.1 to 2.2.5 describe selection criteria, evaluation methods, and specific tool evaluations in various systems engineering domains. Lists of available tools are described in section 2.2.6. Section 2.2.7 and 2.2.8 summarize evaluations of specific tools. Techniques for evaluating and ranking tools are summarized and described in section 2.2.9. Section 2.2.10 summarizes the related work and introduces the contribution to the field of this thesis.

2.2.1 Vendor selection

Weber et al. [6] reviewed 74 articles concerning vendor selection in the context of industrial purchasing and identified 23 selection criteria, of which some are also important for selecting systems engineering tool vendors. Criteria for tool capabilities are extremely important for tool selection but are out of scope of their work and were therefore not included.

2.2.2 Requirements management tool selection

Requirements management (RM) is a major part of SE and may benefit from using dedicated tools. Several researcher have investigated RM tool selection.

Wiegers provided a comparison of four popular RM tools [7] and a weighting-based method to select a tool [8, chp. 21].

Hoffmann et al [9] presented a comprehensive catalog of requirements for RM tools according to different user roles.

Zainol and Mansoor [10] defined a set of elements that requirements management tools must provide, and they conducted a comparative study of 10 tools. They concluded that no tool provided all the elements they required.

Algazzawi et al. [11] made a comparative study of four popular requirements management and systems modeling tools for development of secure enterprise software. The methods that were applied, the evaluation criteria and the sources of data were not reported.

Schwaber and Sterpe [12] discuss the important top-level factors to include and exclude concerning requirements management. Their recommendation is to aim for application life-cycle management (ALM) tools instead of stand-alone RM tools. Both ALMs and product life-cycle management (PLM) tools were considered as candidates in the cases in section 2.6 and 2.7.

The evaluation of specific tools are mostly obsolete because quality and availability of relevant tools have evolved significantly since the cited works were published. However, several of the recommended evaluation criteria were utilized as inspiration for the questionnaire that was used during the development of the tool specification in the case of selecting a traceability tool (section 2.6).

Gotel and Mäder [13] promote a problem-oriented approach to select requirements management tools as part of a wider solution that also includes management of processes, people, expectations, etc. Later [14], they presented a 7-step guide to selecting traceability tools, which included steps to optimize processes to benefit maximally from new tools. The 7 steps (agree on problem and terminology; understand the problem and commit to tackling it; identify stakeholders; determine requirements and constraints; design the wider requirements management system; assess and select tools; plan for tool introduction, adoption and ongoing use) provide a solid foundation for selecting RM tools and can easily be adapted for general selection of SE tools. After the adaption and inclusion of more details and additional steps, we found it more practical to divide the steps into 4 major steps with several novel sub-steps as described in section 2.3.

Matulevičius et al. [15] reported an experiment where 4 groups of students evaluated 2 goal modeling tools using 2 different evaluation approaches. An interesting finding was that the best quality of the goal model was achieved with the tool and language that were rated lowest. Thus, extremely precise evaluating and ranking do not guarantee optimal tool selection.

2.2.3 Modeling tool selection

Systems modeling is becoming increasingly important in SE due to the continual growth of systems complexity. Several researcher have investigated MBSE tools.

Bone and Cloutier [16] conducted a survey on the state of MBSE in 2009 with a focus on SysML and identified a set of inhibitors to the adoption of MBSE. Resistance to change, lack of management support, startup costs, and modeling tools were the worst inhibitors.

Le Blanc and Korn [17] have presented a structured approach to screen, evaluate, assure and finally select computer-aided systems engineering (CASE) tools. The requirements for the tools must be developed prior to using this method.

Estefan [18] describes some popular model-based systems engineering methodologies, languages and tools. The survey focuses on methodologies rather than languages and tools. The survey is purely descriptive and no selection criteria are provided.

Friedenthal et al. [76] and Algazzawi et al. [11] also discussed selection of modeling tools.

However, we did not find any common understanding of what MBSE includes. For some, it only facilitates communication between stakeholders. For others, it includes systems simulations, code generation, modeling of the entire enterprise of the organization, etc. Different tools are required for such varied purposes. We therefore developed a classification of the MBSE disciplines. This classification was utilized for defining proper evaluation criteria in the case of selecting modeling tools (section 2.7).

2.2.4 Simulation tool selection

Verification of system models by running simulations are becoming increasingly important as complexity grows. Several researchers have investigated simulation tool selection.

Banks [19] has presented a method for selecting simulation software, where the number of tool candidates is reduced to a manageable set by evaluating the input, processing, output, environment and cost features using a weighted scoring model. The evaluation is based on already available information. Only the tools that pass the first screening are tried and tested.

Benjamin et al. [20] have presented an expert system for easing the selection of simulation tools. The user enters data concerning the simulation problem and the tool returns a list of applicable tools.

Greenstein [21] has discussed selection criteria for selecting computer-aided engineering (CAE) simulation tools for ASIC designs with a focus on standards and synchronized simulators.

Davies and Williams [22] used the analytic hierarchy process to evaluate and select manufacturing simulation software for an engineering company. After determining the best tool out of five, the weights were changed to achieve a more desirable long-term solution.

Nikoukaran et al. [23] made a survey of methodologies, evaluation techniques, criteria and recommendations for selecting simulation software in manufacturing. The survey identified a lack of a unified approach. Later, Nikoukaran and Paul [24] developed a hierarchical criteria framework for evaluating simulation software where new criteria can be added to the hierarchy as needed. Their framework is an attempt at a unified approach.

Azadeh and Shirkouhi [25] describe a fuzzy analytical hierarchical process for evaluating simulation software to reduce the effects of uncertainties associated with selection criteria.

Attia et al. [26] conducted a study to identify selection criteria for simulation tools for building architects and engineers. From a list of 389 tools, 10 were selected for evaluation. The study showed a wide gap between architects' and engineers' selection priorities and ranking of tools.

Ereeş et al. [27] presented an application of the analytical hierarchy process (AHP) for selecting simulation software in education based, where 20 criteria in 6 groups were compared pairwise by 13 academic experts. Consistent comparisons were made by 5 experts only. They did not evaluate any tools but demonstrated the use of AHP to prioritize criteria. Simulation is just one of the parts of MBSE, so the criteria that are used in the cases in this thesis are much more comprehensive and reflect the actual needs of real stakeholders.

Franceschini et al. [28] conducted a survey of frameworks using discrete event system specification and defined 8 selection criteria. 7 out of 12 frameworks were evaluated, and performance analyses were made for 3 frameworks. No selection method was reported.

Robinson [29, chp. 3] described some popular simulation software and how to select among alternatives based on a 5-step process and a set of possible evaluation criteria.

Damij et al. [30] presented a hierarchical decision model to selection of business process simulation software (BPSS) based on a novel approach. They made detailed investigations for 33 out of 120 available BPSS tools and found similar results as in a previous analysis, thus verifying their method.

Rashidi [31] conducted a survey on taxonomies for simulation software. 6 taxonomies were defined and 62 simulation software were categorized according to the taxonomies. Detailed criteria and methods were outside the scope of the article.

The evaluations of specific tools in the cited works were not sufficiently detailed to clarify the capabilities of the tools. However, several of the recommended evaluation criteria were used as inspiration for developing requirements for the MBSE disciplines in the case in section 2.7. Several steps of the recommended methods were also used as inspiration for the method that is proposed in section 2.3.

2.2.5 General software selection

Many modeling and simulation tools used in SE utilize tool chains that are built on top of existing software frameworks. Selection of general software is therefore increasingly important in SE.

Jadhav and Sonar [32] conducted a comprehensive survey of methods, techniques, criteria and support tools for selecting software packages. They identified a lack of a generic approach and proposed future work to define a common framework. Later [33], they presented a framework consisting of a generic methodology, evaluation criteria and metrics, and a hybrid knowledge based system (HKBS) (i.e. expert tool) to assist decision-making. The advantages of HKBS were demonstrated. However, their case showed that AHP, WSM and HKBS produced the same ranking with almost identical relative scores. However, WSM was shown to be easier to calculate: For M alternatives and N criteria AHP requires M*(M-1)*N/2 pairwise comparisons, whereas WSM only requires M*N. Their work influenced our choice of evaluation method as described in section 2.2.9.

Sarrab and Rehman [34] defined quality characteristics to evaluate and select open source software, and they applied them to the selection of network tools and learning management systems. However, they defined several selection criteria of general interest such as quality (availability, reliability, performance, usability, functionality, maintainability, reusability, testability, security, documentation, level of support), intention of use, user satisfaction, and net benefits.

2.2.6 Lists of available tools

The websites of Alexander [35], Ludwig Consulting Group [36], and the Volere project of Atlantic Systems Guild Ltd [37] provide comprehensive lists of tools related to management of requirements, life-cycles, etc. These lists were used to identify tool candidates in the case of selecting a traceability tool (section 2.6). Weilkiens' [38] website provides a list of popular SysML modeling tools. These tools were considered as candidates in the case of selecting modeling tools (section 2.7).

2.2.7 Evaluation of specific tools

Specific tools were evaluated or discussed in several works. Tools that were evaluated in [7], [12], [15], and [10] were considered as candidates in the case of selecting a traceability tool (section 2.6). Tools that were evaluated in [17], [26], [28], and [34] were considered as candidates in the case of selecting modeling tools (section 2.7). The tool evaluations in [11] [31] and [30] were published too late for the cases in this thesis but are included here for completeness.

2.2.8 Heterogeneous systems modeling tools

Ptolemy-II (PtII) is an open source simulation tool that currently supports 22 domains or modes of computation (MoC) [59]. With PtII, it is possible to combine the various domains in more or less arbitrary order. PtII takes care of synchronizing the simulation of the different domains, so the modelers are not required to have strong knowledge of the underlying technology. However, the modelers must have basic knowledge of the different domain and MoC to obtain valid simulation results. PtII is widely used in education and academia, but it seems not to have been widely adopted by the commercial industries.

The Simulink [74] product family from MathWorks contains numerous tools for heterogeneous systems simulation. The core tool, Simulink, supports the essential MoCs [60] and [61]. Additional tools such as Stateflow [75] and SimEvent [62] increase the number and diversity of possible MoCs. MathWorks provide numerous toolboxes and specialized features. Simulink is therefore widespread and may be considered as industry standard.

Destecs [54] is an open-source modeling tool that integrates the 20sim simulation engine [53] with the modeling language used in the Vienna development method (VDM). Modeling in Destecs is different from modeling in PtII or Simulink. The textual VDM models have no graphical views in Destecs so modeling resembles source code programming. VDM is used for the discrete event (DE) domain. Other domains are modeled in the 20sim tool. With 20sim, models are entered graphically as with PtII and Simulink. The modeler is responsible for defining the contract between the VDM simulator and the 20sim. The contract must contain a list of shared and monitored variables that are used to synchronize the two simulators.

With Modelica [57] and [58], it is possible to mix the different MoC domains implicitly, whereas most simulation tools require explicit separation of the domains.

Heterogeneous systems may be modeled and simulated with other tools, such as Mentor Graphics SystemVision [55], variants of SystemC, National Instruments LabView [56], UPPAAL [52], etc.

2.2.9 Evaluating and ranking tools

Different qualitative and quantitative methods have been proposed and utilized for ranking and selecting tools.

Qualitative methods based on feature analysis were discussed in [17], [7], [9], [12], [33], [10], [11], and [28]. Such methods are easy to apply but they cannot be used for ranking because no values are provided for weights, scores and priorities.

Quantitative weighting scoring methods (WSM) were discussed in [6], [19], [23], [7] [8, chp. 21], [15], [32] [33], and [34]. WSM methods are generally easy to use.

Quantitative methods based on various variants of Saaty's [39] analytical hierarchy process (AHP) were discussed in [22], [23], [24], [25], [32] [33], and [27]. Expert systems to assist in selection of tools were proposed and discussed in [20], [30], and [32].

The AHP methods are sometimes claimed as being superior to WSM, because (1) the pairwise relative comparisons used in AHP are assumed to provide better judgments than the absolute ratings used with WSM, and (2) the consistency of the evaluations can only be calculated with the AHP method. However, in [33] it was shown that WSM produced results that were comparable to the results from AHP and similar methods, but with fewer calculations.

Quantitative tool ranking is a case of multi-objective optimization (MOO). Marler and Arora [40] analyzed and compared a large range of MOO methods including the methods discussed above. Other methods may also be applicable to tool selection, and they can perhaps produce results that are more precise. However, more precision does not guarantee optimal tool selection as shown in [15]. In this thesis, the simplest form of WSM to rate and rank tools is chosen, because it offers ease of use and sufficient precision.

2.2.10 Related work summary

In tool selection situations, there are generally two cases: (1) Select the best tool and accept its shortcomings. (2) Select a set of complementary tools that fulfills all requirements. The cited works only concern with the first case, where it is also assumed that there is a clear winner (highest score). None has addressed the situation where there are several tools with almost identical scores, and none has considered the need for gap analyses to avoid selecting the wrong tools. Our method handles both case (1) and (2). For case (1), it includes gap analysis to avoid selecting tools with essential shortcomings. For case (2), it utilizes set theory to identify feasible sets of tools that meet all (essential) selection criteria.

Most cited works concern requirements management or simulation tool selection based on generic criteria frameworks and evaluation of features as claimed by tool vendors. Tool comparisons in older literature may be obsolete or incomplete because the availability and quality of such tools change rapidly. In this thesis, selection criteria were derived from both from generic frameworks (literature) and from actual needs of real stakeholders and a large collection of tools were evaluated by conducting experiments rather than relying on claimed qualities.

We were unable to identify any comprehensive literature on the selection of tools for general SE, e.g. MBSE tools. The classification of MBSE disciplines that is proposed in section 2.4 made it possible to select a set of modeling tools to implement MBSE in an industrial setting. The method that is proposed in section 2.3 can be generalized to selection of other types of SE tools by substituting evaluation criteria.

No cited works reported how the selected tools actually performed in industrial, academic, educational or similar organizations. In this thesis, a discussion and evaluation of the tools that were selected for a manufacturer of medical embedded systems are presented.

2.3 Proposed selection method

The method for selection of systems engineering tools proposed in this section is inspired by several sources as described in section 2.2. Several steps were added to selected steps of related methods to accommodate more scenarios and to remedy severe shortcomings of previous methods. Most significantly, gap analyses to prevent selecting infeasible tools and techniques to select tools for both single- and multi-tool scenarios were added. The resulting steps were divided into 4 groups corresponding to 4 major phases of change projects. The major steps are shown in figure 2.2 and outlined below with their associated activities:

1. Justify and initiate change project by:
    - Identifying the core problem.
    - Demonstrating the need for new tools.
    - Developing a business case.
    - Considering the implications of introducing new tools.
    - Analyzing benefits beyond solving the core problem.
    - Identifying and addressing potential obstacles.
2. Develop specification and evaluation criteria by:
    - Identifying stakeholders including potential users.
    - Interviewing stakeholders and optionally observing how potential users currently perform their work.
    - Analyzing answers and observations.
    - Specifying requirements and constraints.
    - Approving specifications with stakeholders.
    - Developing selection criteria and priorities from the specifications.
3. Investigate tools and select best candidate(s) by:
    - Scanning the tool market to get a comprehensive list of candidates.
    - Screening candidates based on claimed features, pricing, overall impressions of the tools and vendors, etc.
    - Installing, trying and evaluating likely tool candidates.
    - Rating all evaluation criteria for each tool.
    - For single-tool solutions: Ranking the tools by applying WSM, AHP or similar methods, and selecting the highest-ranking tools for further analysis (see section 2.3.1 below).
    - For multi-tools solutions: Identifying feasible sets of tools that meets all essential requirements, followed by ranking and selecting the best sets of tools for further analysis (see section 2.3.2 below).

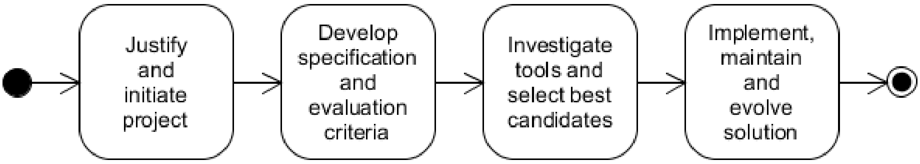

Justify and initiate project
Develop specification and evaluation criteria
Investigate tools and select best candidates
Implement, maintain and evolve solution

Figure 2.2: Major steps of proposed method for selecting MBSE tools.

- Analyzing the gaps or non-conformance for the (set of) candidates and selecting the best tool (set).

4. Implement, maintain, and evolve solution by:

   - Installing, configuring and customizing tools.
   - Adjusting processes and optionally adjusting the organization to benefit maximally from the introduction of new tools.
   - Developing user guides and conducting training.
   - Migrating existing data to new tools.
   - Conducting pilots followed by phased rollout.
   - Maintaining and evolving tool solution to accommodate new needs or possibilities.

This list of steps and sub-steps suggests a strict sequential process but many activities can be conducted concurrently. The ordering of the activities may also be slightly modified. As an example, it may be beneficial to consider the activities of adjusting the processes and organization from step (4) earlier than shown here. Finally, it may be necessary to iterate through the activities several times, as new information becomes available.

The ranking method for single-tool scenarios is described in section 2.3.1 and the selection method for multi-tool scenarios is described in section 2.3.2. Section 2.3.3 describes the part of the method that concerns gap analyses. The prerequisites of using the method are presented in section 2.3.4.

2.3.1 Single-tool ranking method

Ranking of tools can be achieved by using WSM, AHP or similar methods. Here, we have use a simplified WSM where the rank (R) of each tool are calculated as:

$$R = \sum_{i \in [1..N]} W_i \cdot Si \tag{2.1}$$

Here, N is the number of criteria. W_i and S_i are the weight and the score of the i'th criteria, respectively.

The rank should also be calculated for each level of importance (e.g. mandatory, desired, and optional requirements) to facilitate the gap analyses, see section 2.3.3.

2.3.2 Multi-tool selection method

For multi-tool solutions, a set of complementary tools that meet all essential criteria is needed. For each essential criteria, one composes a set of technologies (modeling methodology, language and tool) that meet the criteria to a satisfying degree. The following shorthand notations, operations and rules are used:

$$\begin{aligned} S_i &= \text{Set that meets criteria } i && (2.2)\\ \{A \textit{ or } B\} &= \text{Set requiring either } A \text{ or } B && (2.3)\\ \{C \textit{ and } D\} &= \text{Set requiring both } C \text{ and } D && \\ &= \{C\} \cup \{D\} && (2.4) \end{aligned}$$

S_i in eq. (2.2) is the set of technologies that meet criteria number i. $\{A \textit{ or } B\}$ in eq. (2.3) is a set where either technology A <u>or</u> B are needed to meet the criteria. $\{C \textit{ and } D\}$ in eq. (2.4) is a set where both technology C <u>and</u> D are needed to meet the criteria. The total set S_{tot} that meets all N criteria equals the union of sets S_i that meet the i'th criteria:

$$S_{tot} = \bigcup_{i \in [1..N]} S_i \tag{2.5}$$

For example, one may have three sets of tools supporting three groups of criteria:

$$\begin{aligned} S_1 &= \{\{A \textit{ or } B\} \textit{ and } \{C \textit{ or } D\}\}\\ S_2 &= \{\{A \textit{ or } B\} \textit{ and } \{E \textit{ or } F \textit{ or } G\}\}\\ S_3 &= \{\{H\} \textit{ and } \{E \textit{ or } J\}\} \end{aligned}$$

The total set required for this example is thus:

$$\begin{aligned} S_{tot} &= S_1 \cup S_2 \cup S_3 \\ &= \{\{A \; or \; B\} \; and \; \{C \; or \; D\} \; and \\ &\quad \{E \; or \; F \; or \; G\} \; and \; \{H\} \; and \; \{E \; or \; J\}\} \end{aligned}$$

This set can be reduced to the following two types of solutions:

$$\begin{aligned} S_{tot1} &= \{\{A \; or \; B\} \; and \; \{C \; or \; D\} \; and \; \{H\} \; and \; \{E\}\} \\ S_{tot2} &= \{\{A \; or \; B\} \; and \; \{C \; or \; D\} \; and \; \{H\} \; and \; \{J\} \; and \; \{F \; or \; G\}\} \end{aligned}$$

Further reduction yields the following 12 potential solutions:

$$\begin{aligned} S_{tot1,1} &= \{A \; and \; C \; and \; E \; and \; H\} \\ S_{tot1,2} &= \{A \; and \; D \; and \; E \; and \; H\} \\ S_{tot1,3} &= \{B \; and \; C \; and \; E \; and \; H\} \\ S_{tot1,4} &= \{B \; and \; D \; and \; E \; and \; H\} \\ S_{tot2,1} &= \{A \; and \; C \; and \; E \; and \; F \; and \; J\} \\ S_{tot2,2} &= \{A \; and \; D \; and \; E \; and \; F \; and \; J\} \\ S_{tot2,3} &= \{B \; and \; C \; and \; E \; and \; F \; and \; J\} \\ S_{tot2,4} &= \{B \; and \; D \; and \; E \; and \; F \; and \; J\} \\ S_{tot2,5} &= \{A \; and \; C \; and \; E \; and \; G \; and \; J\} \\ S_{tot2,6} &= \{A \; and \; D \; and \; E \; and \; G \; and \; J\} \\ S_{tot2,7} &= \{B \; and \; C \; and \; E \; and \; G \; and \; J\} \\ S_{tot2,8} &= \{B \; and \; D \; and \; E \; and \; G \; and \; J\} \end{aligned}$$

Choosing between these alternatives requires further analysis. To select the best alternative, we must choose the set with the highest rank. The ranking method from section 2.3.1 is thus applied to the complete set of tools (instead of to the individual tools). A gap analysis completes the selection method for multi-tool solutions.

The multi-tool selection method was used in the case in section 2.7. The details of the analysis are presented in section 2.2 in [2].

2.3.3 Gap analysis

Ranking tool candidates is not sufficient to find the best alternative. Tools with high rankings may lack essential features that disqualify them as candidates. Tools with lower scores are more applicable in such cases.

An obvious solution to this problem would be to adjust the weights of the requirements to produce a more correct and fair ranking of the tools. However, adjusting of weights to get a feasible solution is time consuming and this approach does not guarantee a solid solution without inspected all requirements after each adjustment. Thus, it will be time consuming and error prone.

Instead, we propose using simple forms of gap analyses. Such analyses can easily be obtained from existing data, if the scoring and ranking of tools are calculated for each level of importance as mentioned in section 2.3.1.

Table 2.4 on page 64 shows an excerpt of a tool evaluation where the rankings are calculated separately for mandatory, desired and optional requirements. The gap analyzes are obtained simply by using filters within the spreadsheet. An example of conformance gaps to mandatory requirements is shown in table 2.5 on page 65. Table 2.6 on page 65 shows an example where unfulfilled "desired" requirements are listed. The tables produced by such filtering can be further processed to produce more "user friendly" presentations of the gaps. Comparing the gaps of different tools on a single presentation was very informative in this case (but not shown in this thesis).

As mentioned in section 2.3.2, gap analyses should also be used in multi-tool scenarios. However, the gaps are not analyzed for individual tools in such cases but for the entire set of tools in each potential solution.

2.3.4 Prerequisites

Using the proposed method to select modeling tools requires an understanding of the types of systems under consideration, because different technologies and methods are needed for modeling different types of systems. Section 2.4 introduces classification of systems.

Knowledge of different modeling disciplines are similarly required, because they address different needs and objectives. Section 2.5 introduces classification of modeling disciplines.

2.4 System classification

Systems can be divided into many different groups depending on the context in which the classification is to be used. Here, the discussion will be limited to classifications that are of significance for the model-based systems engineering activities.

When value propositions are considered, systems can be classified as services, commodities or combinations hereof. Services are often modeled with the Business Process Modeling Notation (BPMN), Business Process Execution Language (BPEL), Unified Modeling Language (UML) or similar modeling languages that allow the systems engineer to model the processes of the services. The optimum choice of modeling methodology, language and tools depends on the nature of the service. Services therefore need further classification into sub categories such as web services, postal services, flight service, etc. Commodities must similarly be classified further to find an optimum set of modeling methodologies, languages and tools. Commodities can be even further classified by the nature of their composition.

Simple systems are typically easier to model than complex systems. Homogeneous systems are typically easier to model than heterogeneous systems. Complex heterogeneous systems often contain many kinds if hardware (analog and digital electronics, mechanical parts, electro-acoustic elements, biology, human operators, etc.) and several kinds of software (operating systems, system logic, user interfaces, digital signal processing algorithms, mathematical computations, etc.). Modeling such systems is significantly more difficult than modeling homogeneous systems that only contain a few equivalent domains.

Special methodologies might be needed for modeling systems that contain harmful elements such as microwave radiators, toxic chemicals, radioactive sources, infectious biology, etc. For safety critical systems, it will probably be necessary to model the quality properties that are needed for executing safety analyses such as fault trees, error propagations, failure mode and effects analysis (FMEA), etc. Other quality aspects may be more important for other kinds of systems. Detailed modeling of timing, performance and reliability is highly relevant for real-time systems. Security (protection against unintended use of the system) may be very important for some systems.

Only one mode of computation is needed for simulating homogeneous systems, whereas the simulations of heterogeneous systems require several modes of computation such as continuous time (CT), discrete events (DE), finite state machines (FSM), heterochronous dataflow (HDF), etc. For a list of supported domains in the Ptolemy-II modeling tool, see [59]. Other tools may support different domains or modes of computation as discussed in section 2.2.8.

When business models are considered, systems can be divided into groups consisting of purchased, rented or subscribed products. Some business models include advertising while others do not. Some business models are based on open source strategies while others are not. The business model can have big impact on the required modeling activities, because it can affect the requirements concerning system behavior, architecture, performance, documentation, etc.

Usage scenarios of systems can also have a big impact on the required modeling. Throw-away-systems that are only used once may require less modeling than systems that are (re)used in different many contexts, by many different kinds of users, or as parts of other systems.

The major classification criteria that should be considered for successful modeling of embedded devices are:

- Modeling complexity:
  - Systems that can be modeled with a single modeling language and tool.
  - Systems that require several modeling languages and tools to provide complete and accurate descriptions of system behavior, performance, etc.
- Simulation complexity:
  - Systems that can or shall be analyzed with formal methods.
  - Homogeneous systems that can be simulated with a single mode of computation.
  - Heterogeneous systems that require several modes of computation for successful simulation.

Figure 2.3 shows examples of keywords that can be used to classify embedded and smart systems. These keywords were used to describe the characteristics of hearing systems in section 1.4. The characteristics of typical embedded systems and smart systems are further discussed in section 3.2 and 4.2, respectively.

2.5 Modeling disciplines

Informal conversations with various practitioner and specialists involved in MBSE have revealed huge differences in their perception of the nature and objectives of MBSE.

For some experts, MBSE is just considered as the modeling activities that formalizes SE. For other experts, MBSE is mainly concerned with a small subset of SE, e.g. defining use cases, interfaces or system structure. For many others, MBSE is expanded into the software engineering domain by focusing on e.g. code generation with

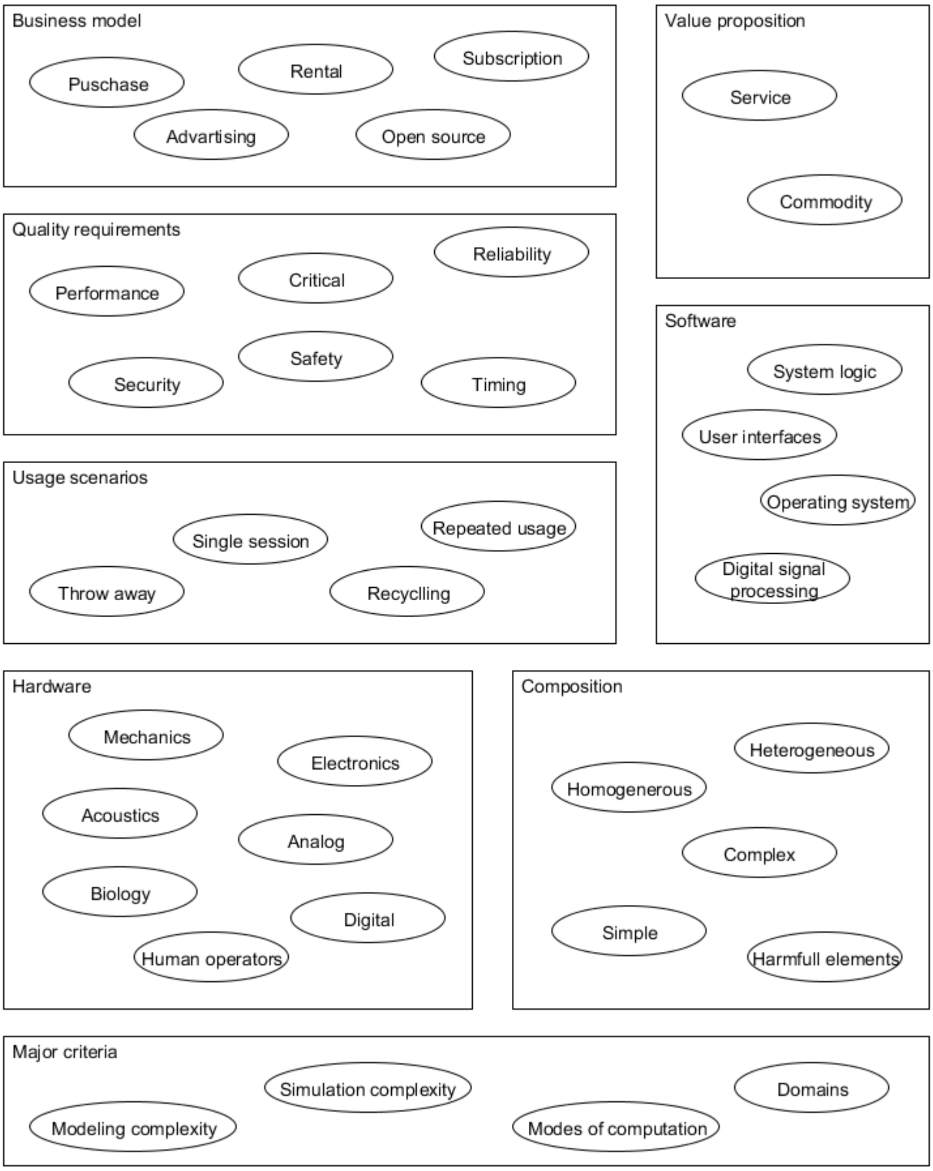


Figure 2.3: Examples of system classification keywords.

model-driven architecture (MDA). Finally, some consider MBSE as the discipline of modeling the entire enterprise of the concerned organization.

Presenting various "definitions" of MBSE to some practitioners did not streamline the perceptions. Without a common understanding, it is very difficult to determine the correct requirements for the modeling tools. Clearly, a categorization of the various sub-disciplines of MBSE is needed to let organizations accurately determine the needs by asking questions that are more precise.

Searching existing literature did not reveal any satisfactory taxonomy. Therefore, we propose the following classification:

- Fundamental modeling (FM) as described in section 2.5.1.
- Behavioral simulation modeling (BSM) as described in section 2.5.2.
- Architectural analyses modeling (AAM) as described in section 2.5.3.
- Architecture/behavior co-modeling (ABCM) as described in section 2.5.4.
- Coherent modeling (CM) as described in section 2.5.5.
- Integrated engineering (IE) as described in section 2.5.6.
- Enterprise modeling (EM) as described in section 2.5.7.
- Other engineering activities as described in section 2.5.8.
- Other activities beyond engineering as described in section 2.5.9.

This classification is based on elements that should be familiar to experienced systems engineers even though it may not be agreed upon by the MBSE communities (e.g. INCOSE). However, it provides a sufficient framework for the proposed method. It may be argued that CM and IE are enablers for model consistency rather than true modeling disciplines. However, CM involves application of meta models, and EI integrates modeling with other engineering activities. The other activities may also include forms of modeling. Therefore, the term "modeling discipline" is used for all the classified disciplines even though it may not be entirely correct to do so.

Figure 2.4 illustrates the relationships between the various sub-disciplines. Notice that there can be several BSM, AAM and ABCM models for each FM. CM facilitates consistency between these different models. IE facilitates consistency between models and other engineering activities. EM facilitates consistency across all activities of the companies.

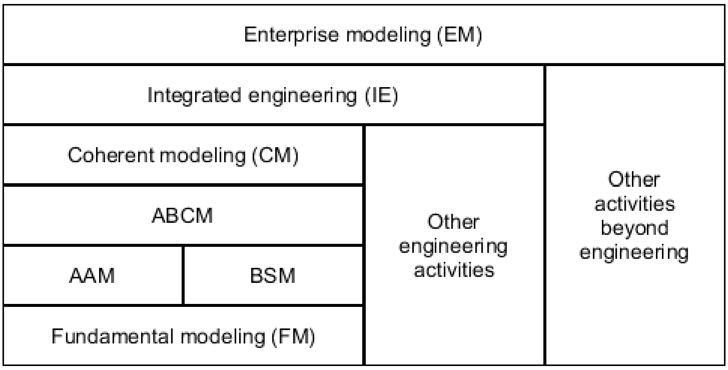

Enterprise modeling (EM)
Integrated engineering (IE)
Coherent modeling (CM)
ABCM
AAM
BSM
Fundamental modeling (FM)
Other engineering activities
Other activities beyond engineering

Figure 2.4: Modeling disciplines.

The characteristics of each discipline must be understood thoroughly in order to successfully use the presented method, because each classified element gives rise to numerous questions that allow the analyst to correlate the needs and available technologies. Sections 2.5.1 through 2.5.7 are therefore dedicated to describing the characteristics of each modeling discipline. Modeling guides for parts of FM, AAM, BSM, CM and EM are available in chapter 2 in [1].

2.5.1 Fundamental modeling

Fundamental modeling (FM) formalizes the fundamental system engineering activities and makes it possible to model life-cycle phases, stakeholders, requirements, constraints, use cases, behavior, and architecture for the system under consideration (SUC), its environment and users. FM, as defined here, focuses on facilitating the communication and collaboration between stakeholders to collect and analyze all ideas, requirements and constraints of the SUC. The modeling rigor needed for automated analyses and simulations is considered outside the scope of FM. Fundamental models typically include at least some of the model elements listed in table 2.1. Modeling guides for parts of FM are available in section 2.2 in [1].

SysML [46] is a very versatile language for fundamental modeling. It can be learned from various sources including the books [47], [76] [48], and [49]. SysML is supported by several commercial and open source tools. With many modeling tools, it is possible to combine SysML and other modeling languages into a single model, which increases the overall expressiveness. Other modeling languages such as UML [50] or MARTE [51] may be used as alternatives to SysML in some cases. In the following

Table 2.1: Typical elements of fundamental modeling.

Elements	Purpose and content
Phases	The lifecycle model describes all phases of the SUC and ensures that they are considered during the early analysis and design phases.
Stakeholders	The stakeholder model describes all stakeholders involved in the project including users of the SUC, and it ensures that all important stakeholders and their concerns, needs, requirements and constraints are considered.
Requirements	The requirement model describes the requirements of all stakeholders as well as the constraints of the project and the conditions relating to the SUC and its environment.
Use cases	The use case model describes the functions that the SUC shall provide during its lifecycle phases.
Test cases	The test case model describes the test cases the SUC shall be subjected to as part of the verification and validation process.
Behavior	The behavioral model describes how the SUC reacts to external and internal stimuli. It therefore describes which system outputs are generated for a given set of system inputs, environmental conditions, and internal states such as malfunctions of its components. Thus, it gives detailed information about wanted as well as unwanted behavior. Behavior is typically modeled as activities, sequences or state machines.
Structure	The structural model describes the system domain, context and architecture. The domain describes the hierarchy, properties and interconnections of the blocks that represent the SUC, the environmental systems and the users. The system architecture similarly describes the hierarchy, properties and interconnections of the blocks that represent the SUC and its sub-systems and components. Architectures are typically categorized as functional, logical or physical. The most important metrics to characterize architectures concern the number and structure of building blocks and the resulting interfaces and dependencies.
Data	The data model describes the data types of information and physical items that are used in the system and the domain.
Traceability	Traceability is an important aspect of fundamental modeling to obtain healthy designs by ensuring that all model elements can be traced to the originating requirements and stakeholders. Traceability is obtained by associating related elements.

description of the elements from table 2.1, it is assumed that SysML is used for FM.

The system life-cycle is modeled to ensure that all phases are represented in the model. SysML has no special means for modeling of the system life-cycle, but it can be modeled as SysML requirements that are stereotyped «lifecycle» or similar.

Stakeholders are modeled to ensure that all important stakeholders and their concerns, needs, requirements and constraints are represented. Stakeholders are typically modeled on SysML use case diagrams. Both project influencers and system users are modeled.

Textual requirements and constraints can be modeled on requirements diagrams or in tables in a SysML requirements model. Many organizations prefer to use dedicated requirements management tools, because such tools offer many features that are absent in SysML modeling tools, e.g. version control, automated history logging, etc. In such cases, the requirements elements must be mirrored in the SysML tool to allow traceability to other modeling elements. Several tool vendors have solutions for synchronizing SysML requirements and requirements stored in other tools.

Use cases describe the functions provided by the SUC, and their relationships to the users and environmental systems. They are typically modeled on SysML use case diagrams.

Test cases are modeled to ensure that important aspects of the system are tested correctly, and they can be modeled on either use case diagrams or requirements diagrams.

Behavioral models describe how the SUC reacts to external and internal stimuli. It therefore describes which system outputs are generated for a given set of system inputs, environmental conditions, and internal states such as malfunctions of its components. Thus, it can give detailed information about wanted as well as unwanted behavior. Activity, sequence or state machine models are used to describe system behavior in SysML. Other languages such as UML or MARTE offer additional behavioral modeling constructs that can be used if the behavior cannot be expressed in SysML. The fundamental behavioral models may be refined into the behavioral simulation models as described in section 2.5.2.

Structural models describe the system domain or context as well as the system architecture. The domain describes the hierarchy, properties and interconnections of the blocks that represent the SUC, the environmental systems and the users. The system architecture similarly describes the hierarchy, properties and interconnections of the blocks that represent the SUC and its sub-systems and components.

Good architectures are developed in several stages. Functional architectures describe the system functions or capabilities. Each function is modeled as a «functional» block. The logical architecture is developed from the functional architecture by adding the «logical» blocks that represent internal system functions such as fault detection, database maintenance and similar functions. These behavioral architectures must be decomposed to a degree that allows each functional and logical block to be allocated to exactly one «physical» block from the physical architecture. The functional, logical and physical blocks are thus related by means of allocations. A simplified approach, where behavioral elements (activities, sequences, state machines, etc.) are allocated directly to physical blocks, can be used for simple systems.

The most important metrics to characterize architectures concern the number and structure of building blocks and the resulting interfaces and dependencies. The block structure can be modeled on SysML block definition diagrams (BDD) for all types of architectures. Interfaces or interconnections can be modeled on SysML internal block diagrams (IBD). Other important architectural metrics concern competitive quality properties as well as development cost, development time schedule and required resources, etc.

The data model describes the information and physical items that flow in the system or its domain. Data structures can be modeled as SysML blocks or UML classes.

Traceability is an important aspect of fundamental modeling to obtain healthy designs, where all model elements can be traced to the originating stakeholders and their requirements. Traceability can be modeled by adding ≪trace≫, ≪deriveReqt≫, ≪verify≫, ≪satisfy≫, ≪refine≫, and ≪allocate≫ relationships between the different elements of the model.

Fundamental modeling may also include other elements that make the model more “user friendly”. The model may be structured by using packages. Model elements may be annotated with comments. Special views can be created for individual stakeholders. Etc.

All information about the system is collected into a system model, which is typically contained in a central repository. This repository can be kept in a database that allows many stakeholders to access and contribute to the system model concurrently. A high degree of transparency is therefore attainable. The repository is often machine-readable and can be subjected to automated model checking and other forms of analyses. Humans typically interact with the model via diagrams. It is not uncommon to have separate views and diagrams for different stakeholders.

Section 2.2 in [1] further describes modeling needs (2.2.1), languages and tools (2.2.2) associated with fundamental modeling. Guides for modeling of systems life cycles (2.2.3), stakeholders (2.2.4), system context (2.2.5), requirements and constraints (2.2.6), use cases (2.2.7), scenarios (2.2.8), test cases (2.2.9), and traceability (2.2.10) are also presented in the appendix. A summary of the proposed fundamental modeling techniques is presented in 2.2.11.

2.5.2 Behavioral simulation modeling

Behavioral simulation modeling (BSM) augments the fundamental behavioral models by producing models that can be subjected to automated and interactive simulations. BSM therefore makes it possible to predict system behavior and correct errors before actually building the physical system. Separate behavioral models may be developed for the functional, logical and physical architectures.

The approach for behavioral modeling depends largely on the type of SUC. Homogeneous and heterogeneous systems require vastly different approaches to modeling and simulating. Homogeneous systems can be simulated by using a single mode of computation (MoC), as described in section 2.5.2.1. Heterogeneous systems require several MoCs during simulation (e.g. continuous time (CT), synchronous data flow (SDF), finite state machines (FSM)), as described in section 2.5.2.2. Each MoC typically requires a separate simulator, so simulating heterogeneous systems requires synchronization of several separate simulators.

Simulating behavioral models may be interactive or automated depending on the needs, the capabilities of the simulation tool, etc.

Section 2.4 in [1] further describes needs (2.4.1), characteristics (2.4.2), elements (2.4.3) and benefits (2.4.4) of behavioral modeling. Examples of modeling languages and tools associated with behavioral simulation modeling are also presented (2.4.5). Guides for behavioral modeling of homogeneous systems (2.4.6) heterogeneous systems (2.4.7) with different modeling technologies are presented. A summary of the proposed behavioral modeling techniques is presented in 2.4.8.

2.5.2.1 Homogeneous systems

Homogeneous system models are characterized by requiring only a single mode of computation (MoC) during simulation. Examples of homogeneous systems are (1) pure software systems that can be modeled and simulated with SysML, UML, AADL, etc., or (2) pure digital hardware systems that can be modeled and simulated with VHDL, SystemC or Verilog. The modeling languages and tools used for heterogeneous systems may also be used for homogeneous systems but dedicated domain specific tools may be a better choice.

In many cases, it will be possible to create valid system models in SysML or other UML-based languages such as xtUML or MARTE. Some modeling tools can simulate the models directly inside the tool and visualize the execution by highlighting active actions or states in the diagrams. Other tools may not have this feature but can generate executable simulation code from the models. Generated code is typically compiled and executed in external software development tools such as Eclipse. Successful code generation requires that all behaviors be executed by blocks or classes. In SysML, this is ensured by using «allocate» relationships. However, some code generators do not recognize such relationships but require careful structuring of the model, such that the behaviors can be executed as (or called from) so-called class behaviors.

Homogeneous modeling does not necessarily exclude heterogeneous systems. In some cases, it can be justified to simplify a heterogeneous system into a homogeneous model to take advantage of the many good modeling and simulation tools that are available for homogeneous systems. For an audio system that contains both analog and digital processing as an example, it may be feasible to model the whole system in either the analog or the digital domain. The success of this approach depends on the level of fidelity that is required from the model and the simulation results.

Examples of homogeneous systems modeling are presented in section 2.4.6 in [1].

2.5.2.2 Heterogeneous systems

Heterogeneous systems contain two or more different modeling domains that require different MoC during simulation. In audio processing equipment as an example, the analog domain requires continuous time (CT) computation, whereas the digital domain may use synchronous data flow (SDF) computation, and the system controller may use finite state machine (FSM) computation. Many different domains and MoC have been described and implemented in various tools. Each MoC may require a separate simulation engine. These must be coordinated by using shared variables and time synchronization to obtain valid simulations of the overall system.

Heterogeneous system behavior may be modeled in SysML by using stereotypes such as «continuous», «sdf», «fsm» for the behavioral elements. However, currently SysML modeling tools do not provide engines for simulating or generating code that make such models executable. So alternatives are needed. Section 2.2.8 describes some tools that may be used for modeling the behaviors of heterogeneous systems.

Examples of heterogeneous systems modeling are presented in section 2.4.7 in [1].

2.5.3 Architectural analyses modeling

Architectural analysis modeling (AAM) augments the fundamental structural modeling by producing models that can be subjected to automated and interactive analyses. The fundamental modeling of block hierarchies and interconnections define what is needed to make a system that can provide the desired system functions. To make the most optimal system one also need to model the properties of system components as well as the constraints and conditions under which they apply. The parameters and constraints are related through equations.

The architectural analysis consists of predicting the properties of the SUC:

- System performance, reliability and other quality attributes.
- Parameter sensitivity and impact.
- Faults propagation and impact.
- Risks, cost, etc.

Analyzing the architectural models may be interactive or automated depending on the needs, tool capabilities, etc.

Very simple static equations such as summing of weights can be analyzed by using ordinary spreadsheets or mathematical scripts such as Matlab, Python, etc. However, this approach may not be feasible for dynamic equations that depend on e.g. system modes. In addition, it may be difficult to ensure traceability and consistency from such artifacts to the fundamental system models.

Another option is to use a dedicated architecture modeling language such as AADL, which facilitates easy modeling and analyzing of certain system properties. However, customized analyses may be difficult to obtain with such languages, which may therefore not be the best choice in all cases.

A third option is to use generic simulation tools such as Simulink, Ptolemy-II, UPPAAL, Modelica, etc. High levels of customizations are possible with these tools. However, even the most simplistic analyses requires detailed modeling in contrast to dedicated architectural modeling languages, where many analyses are predefined.

A fourth option is to use a SysML tool that can run architectural analyses by executing parametric models. This option has the advantage that consistency and traceability between FM and AAM can be ensured. However, not all SysML tool support parametric modeling, and even fewer have the ability to execute such models.

The best choice of tools for AAM depends on the actual needs concerning easy standard analyses, customized analyses, parameter sensitivity analyses, or analyses of fault propagations, dynamic architectures, etc.

Section 2.3 in [1] further describes needs (2.3.1), characteristics (2.3.2), elements (2.3.3) and benefits (2.3.4) of architectural modeling. Examples of modeling languages and tools associated with architectural analyses modeling are also presented (2.3.5). Guides are presented for architectural modeling of system hierarchies (2.3.6), interconnections (2.3.7), structures (using AADL) (2.3.8), systems properties (2.3.9), and flow and time (2.3.10) with different modeling technologies. Advanced modeling of errors, dynamic architectures, and real-time operating systems (RTOS) are also discussed (2.3.11). A summary of the proposed architectural modeling techniques is presented in 2.3.12.

2.5.4 Architecture/behavior co-modeling

The architecture and behavior modeling discussed in the previous sections assume that system behavior and the architectural properties are independent and therefore can be seen in isolation. However, architectural resources are often shared such that seemingly independent functions are competing for allocation of resources, which may lead to unforeseen interdependence. In addition, architectural properties are often affected by behavioral elements (e.g., when a user activates or deactivates components of subsystems). Behavioral elements may likewise be affected by architectural properties (e.g. certain features may be disabled at high temperatures). In such cases, it will be necessary to co-simulate the behavior and the architecture.

This kind of co-simulation requires synchronization of simulators for behavioral elements and equation solvers for the architectural elements. Architecture/behavior co-simulation is therefore even more complex than simulation of heterogeneous behavioral systems, which only require synchronizing different simulators but not equation solvers. However, equation solvers can be replaced by simulators in some cases. Section 2.4.7 in [1] shows examples of interdependent architectures and behaviors modeled as heterogeneous systems with Ptolemy-II (example 2.17, 2.18 and 2.19). Heterogeneous modeling with Simulink/Stateflow and VDM/20-sim are also discussed.

Code generation from models may also be classified as architecture/behavior comodeling (ABCM) because the behavioral elements must be allocated to and executed by architectural elements (classes). Section 2.4.6 in [1] includes an example (2.16) of modeling for code generation.

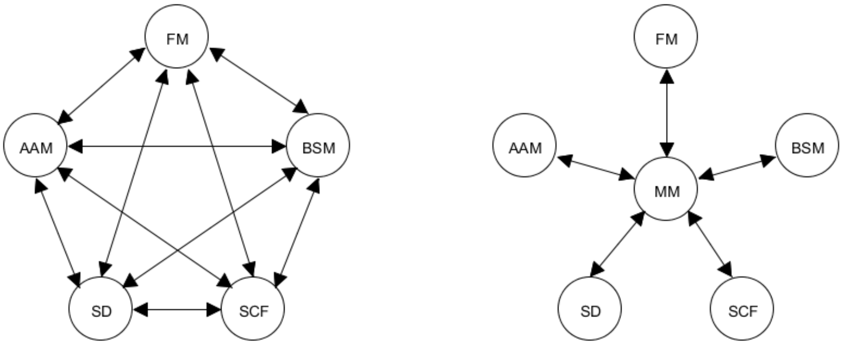


Figure 2.5: Examples of two integration approaches.

2.5.5 Coherent modeling

As mentioned in section 2.5.1, it is important to ensure traceability between the various elements of the models. This can be difficult to achieve if the elements of the models are spread over different modeling tools or expressed in different modeling languages. However, it is even more important in such cases because there is a risk of having undetected inconsistencies between the involved models, which can lead to disastrous errors later. Therefore, a mechanism to ensure consistency and traceability between the various models is needed. Coherent modeling (CM) intends to ensure this by using (semi-) automated model transformations[2].

Figure 2.5 illustrates two different approaches to ensure consistency between models and other artifacts. In this example, 5 types of artifacts are shown as circles: A fundamental model (FM), an architectural analysis model (AAM), a behavioral simulation model (BSM), a set of specification documents (SD), and a set of software source code files (SCF). Transformations are shown as arrows. The left part of the figure shows an approach with point-to-point integration where model transformations are required between all types of models. It can be shown that this requires N*(N-1) transformations between N models. The right part of the figure shows a centralized approach, where a master model (MM) contains all information such that all other models can be generated by model transformation from the master model. It can be shown that this requires 2*N transformations between N slave models and 1 master model.

[2] Techniques to ensure consistency between models and other artifacts are also discussed in appendix 2.5.

Using the centralized approach with master modeling will always be optimal in terms of number of required transformations. Master modeling has other advantages as well. The overall modeling effort is reduced because the properties and interfaces are modeled only once. The tools used for master modeling typically make it much easier to get an overview of the model and to generate informative documents, compared to specialized modeling or simulations tools.

Master models are typically created in SysML, MARTE or similar UML-based modeling languages, where the base language is extended with special profiles for the slave models. The master models must contain all the information needed to generate the slave models, in the form of properties and stereotypes. The slave models for various targets can then be generated by customized plugins that facilitate the model transformations. Several examples of using SysML or MARTE as front-ends for AADL have been reported, e.g. [63]. Examples of using UML as front-ends for Simulink have also been reported, e.g. [64].

Master modeling also has some disadvantages, though. The most important of these is the need for creating and maintaining model transformation tools and meta-models of the slave models. In addition, it may not be possible to represent all aspects of the target language in the master modeling language.

Therefore, alternatives must be identified for those cases. Integrated engineering as described in section 2.5.6 is another approach for reducing the risk of inconsistent models that can be used in cases where coherent modeling is infeasible or impractical.

2.5.6 Integrated engineering

Several modeling languages and tools are often needed to describe all aspects of a SUC. A separate tool may be used for requirements managements. A SysML modeling tool may be used for fundamental modeling. Architectural modeling may be done in AADL, whereas behavioral modeling may be done in Simulink or Ptolemy-II. These models will be used as preparation for implementing physical system artifacts, such as hardware, software, user documentation, manufacturing instructions including bill of materials (BOM), etc. The system information is thus distributed across a large set of interdependent artifacts.

The overall purpose of integrated engineering (IE) is to bring all the separate engineering disciplines into an integrated whole by providing a suitable environment.

Cyber-environments with a central data-base containing all system artifacts (requirements, model elements, source code, hardware design files, bill of materials, etc.) facilitate virtual integration of all disciplines and team members that collaborate concurrently on the artifacts.

Product life-cycle management (PLM) tools are typically used as backbone for IE because they offer features that make it possible to highlight those artifacts that are affected by changes to a requirement, a model or other artifact, thus minimizing (but not eliminating) the possibility of inconsistencies between models and other artifacts. Besides version control and automatic history logging, they usually also offer features for defining base lines, platforms, etc. Traceability can typically be shown as text documents, tables, matrices or as visual graphs.

Several solutions for integrating modeling with PLM tools have been reported, e.g. [65], [66], [67], and [68]. Some approaches use loose integration where artifacts are linked by hyperlinks. Other approaches use tighter integration between the PLM and the modeling tool. Tight integration is implemented either by integrating modeling into the PLM tool or by integrating PLM functions into the modeling tool. Currently there is little evidence regarding the advantages and disadvantages of these different approaches.

In databases, artifacts are represented as data nodes, and relations between artifacts are represented as links. Each artifact is thus part of a net to which it is linked. The whole net can automatically be marked as suspect whenever one of the artifacts are modified. By using queries and visualization techniques, it is therefore easy to identify inconsistent artifacts.

However, full consistency cannot be guaranteed because the actual contents of the artifacts are not analyzed. IE is therefore not as strong as CM, but IE may nevertheless be chosen because it requires fewer resources to implement and maintain (no meta-models and transformation tools are required). Future IE tools may be able to analyze the actual content of the artifacts for stronger consistency checks.

The physical setup and work environment for the team working on the SUC is another aspect of IE that may influence the resulting quality. When working on highly interdependent artifacts, it may be beneficial to bring the domain experts together. The use of a dedicated project room, where the team members are placed in clusters around a central large screen monitor to facilitate both individual and collective work, has been reported, e.g. [69]. Such setups may be desirable but not always be possible. However, with suitable IT tools, it is possible to establish virtual team rooms with many of the same characteristics.

2.5.7 Enterprise modeling

Enterprise modeling (EM) is the practice of modeling the entire enterprise of analyzing, designing, implementing, manufacturing, operating, servicing, maintaining, evolving and finally disposing series of systems or products. EM includes models of the organizational structure, processes, resources and other business information. EM has numerous advantages compared to a document-centric approach. All information can be highly structured and located in a single repository, which makes the information easily accessible. Document generators and dashboards make it possible to present the information in a format that makes sense to the stakeholders. The use of well-defined modeling languages reduces ambiguity. By combining EM with other forms of systems modeling, it is possible to analyze and optimize the relationship between system properties and enterprise constraints such as available development resources. EM may thus be seen as the ultimate goal for modeling.

EM is beyond the scope and capacity for many organizations, including the company for which the analysis was made. EM is therefore not discussed further in this thesis. However, section 2.6 in [1] concerning advanced modeling includes a discussion of using enterprise architecture frameworks and modeling tools to facilitate EM.

2.5.8 Other engineering activities

Modeling as described in the previous sections is part of a larger set of engineering activities that are orchestrated in various phases of developing products and systems. Ideation [70] and design thinking [71] [72] are examples of activities in the early phases. Designing, implementing, testing, and validating hardware and software are examples of activities in the later phases.

Simulating system models with hardware or software prototypes in the loop is another example of cross-cutting engineering activities. Sometimes a domain cannot be modeled properly because it is too complex or because the fundamental laws of the domain are unknown or cannot be described easily in the modeling languages. Physical prototypes can replace low-fidelity models and be included in the simulation of the overall system. Simulating with hardware and/or software in the loop thus provides better predictions of the functionality or performance of the overall system. Additionally, simulating with hardware and/or software in the loop can be used to ensure that prototypes of system parts behave as specified by the models. In these cases, the simulation engines must therefore be able to simulate with hardware and/or software in the loop to give realistic predictions of the system behavior, performance, etc.

2.5.9 Other activities beyond engineering

Researching, manufacturing, marketing, servicing, operating, and dismissing systems are examples of activities that may be considered beyond engineering. Modeling and simulating may also be applied to such activities. For example, some systems may require complex scenarios for service and maintenance, which may be difficult to validate. Building prototypes may be impractical due to costs or other reasons. In such cases, it may be possible to use virtual reality interaction during the specification or validation of the scenarios.

2.6 Case study: Traceability tool (single-tool scenario)

The proposed method was applied to selecting a traceability tools for GN Hearing. The tool should be able to provide traceability between product claims, requirements, test cases, test results, defects, etc. It was decided early to focus on single-tool scenarios, such that all types of traceable artifacts could be managed without complicated tool chains. The steps of the method were applied in the following manner:

Step 1 - Justify and initiate change project:

FDA (food and drug administration, USA) [41]and other authorities require full traceability from requirements to test cases, test results, and potential defects for medical devices. Historically this has been done manually at GN Hearing by entering all required data into spreadsheets. However, the company must show the full history of the product development course and not just the final state. The manual processing consumed many resources and produced few benefits beyond the required traceability. So it was decided to search for a traceability tools.

Introducing a traceability tool was expected to lead to better and more effective requirements management, test management, defect management, task management, and project monitoring. A business case was built based on more than 500 potential users (system engineers, developers, testers, project leaders, quality assurance specialists, etc.) with varying types of access types/licenses. A cost limit for a 5-year scenario was defined as a fundamental constraint. Using the same cost scenario for all tools made it easier to compare different license models.

Step 2 - Develop specification and evaluation criteria:

Approximately 40 stakeholders (potential users, line managers, IT specialists, etc.) were identified as important. Most stakeholders were identified during the interviews. A questionnaire with 27 questions was prepared for the interviews. Most interviews were conducted as face-to-face meetings with individual stakeholders but some were conducted as group interviews, virtual meetings, email correspondence, etc. The interviews produced 232 unique answers that were collecting in a spreadsheet. Answers representing conflicting requirements were clarified with the stakeholder and resolved (whenever possible) or ignored (when the conflicts were unsolvable). The remaining answers were reformulated and thus converted into 176 requirements.

The requirements were prioritized and divided into 22 categories with 4 categories of general (G) requirements and 18 categories of feature (F) requirements, as shown in table 2.2. Requirements were prioritized as either mandatory, desired or optional. The numbers of these requirements are shown in column #MR, #DR, #OR and #AR, respectively. The specification also includes 5 assumptions (e.g. how various functions are moved from previous tools to the new traceability tool) and 7 constraints (conformance to various standards, supported operating systems and web browsers, etc.). The specification was reviewed by all stakeholders and approved by the steering group.

Selection criteria were developed directly from the specifications. The weights of the criteria were set to 5 for mandatory requirements, to 3-4 for desired requirements, and to 1-3 for optional requirements. These criteria were later utilized to rank the evaluated tools.

Step 3 - Investigate tools and select best candidate:

The tool market was scanned for candidates. A list of 133 tools was created by searching the internet and by inspecting websites such as Alexander [35], Ludwig [36], Volera [37].

More than 100 websites of tool vendors were inspected for the first screening of candidates. 33 tool were excluded from the analysis because data could not be obtained within the scheduled period of the investigation. 67 tools were excluded during the first screening because they seemed to lack essential parts of the features that were identified from the specification: management of traceability, requirements, testing, reviewing, reporting, work-flow, etc. 6 tools were excluded after receiving further information from the tool vendors due to license pricing or lacking functionality. 4 tools were excluded after watching demonstrations. 26 tools were installed or tested online. In some cases, our own testing did not confirm the vendors' rather optimistic

Table 2.2: Requirements distribution for traceability tool.
G = General requirements.
F = Feature requirements.
#MR = number of mandatory requirements.
#DR = number of desired requirements.
#OR = number of optional requirements.
#AR = number of all requirements.

Requirements category	Type	#MR	#DR	#OR	#AR
Tool components	G	8	3	3	14
Interfaces	G	2	3	6	11
Usability	G	6	10	2	18
Deployment & usage	G	4	1	1	6
Requirements management	F	6	2	2	10
Implementation & design	F	-	3	2	5
Testing management	F	16	8	4	28
Release & configuration	F	4	-	-	4
Issue & defect management	F	-	3	3	6
Base lining & branching	F	-	3	2	5
Change & workflow management	F	3	3	1	7
Review management	F	2	3	-	5
Version management & control	F	5	2	-	7
Task management	F	-	5	2	7
User management & security	F	3	1	-	4
Dashboards & reporting	F	4	3	1	8
Traceability & analysis	F	8	2	2	12
Export & import round-tripping	F	1	2	1	4
Integration features	F	2	2	4	8
Risk and hazard analysis	F	-	-	2	2
General repository features	F	-	-	2	2
Global glossary features	F	-	-	3	3
Total		74	59	43	176

Table 2.3: Specification conformance and gaps for traceability tools.

	Conformance	Gap / Non-conformance		
Requirements:	All	Mandatory	Desired	Optional
Tool 1	71%	19%	33%	54%
Tool 2	63%	27%	44%	58%
Tool 3	70%	22%	38%	41%
Tool 4	82%	15%	17%	35%

claims. 4 of the remaining tools turned out to be too expensive. 1 tool was excluded due to lack of sufficient evaluation data. The remaining 18 tools were subjected to further analysis.

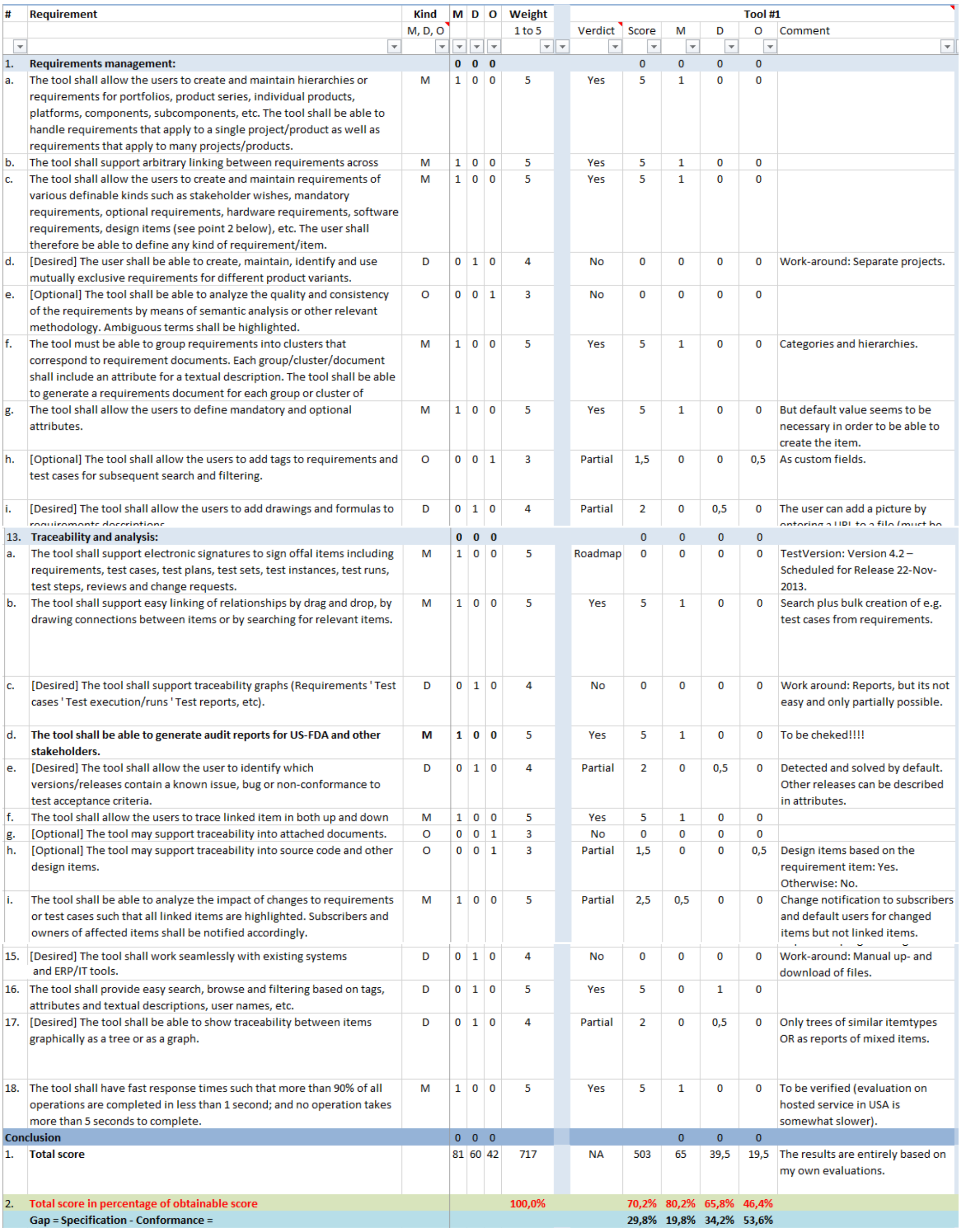
Evaluating all 176 criteria for 18 tools was considered to require too much effort and time. Instead, we evaluated the 18 feature groups to give an overall impression of the tools. 9 tools could not be fully evaluated due to malfunctions or inadequate feature coverage. 4 tools within the acceptable price range were selected for further analysis, which included full evaluations of the 176 criteria. WSM as described in section 2.3.1 was used fourfold: For mandatory, desired, optional, and all requirements. An overview of the summed ratings and gaps is shown in table 2.3. Notice that all tools have gaps in the conformance to even the mandatory requirements. Table 2.4 shows more details of the evaluation of tool #1. Table 2.5 and 2.6 show parts of the gap analysis for mandatory and desired requirements, respectively. Gaps for optional requirements are not shown. The tables 2.4, 2.5 and 2.6 are generated from the same spreadsheet by using different filters for the "kind" and "verdict" columns.

Combining several tools to overcome the gaps was not considered because it would be too difficult to integrate the tools and ensure the required traceability, so in this case we had to select the best tool and accept its shortcomings. Further analysis of the gaps resulted in the exclusion of tool 1 and 2, so the final choice was between tool 3 and 4. Tool 4 achieved the highest score and was deemed easy to introduce but with poor defect tracking capabilities. Tool 3 scored somewhat lower, it but was deemed as a better long-term solution that would be easier to integrate with the company's software development activities. Tool 3 was finally selected after much discussion.

Step 4 - Implement, maintain, and evolve solution:

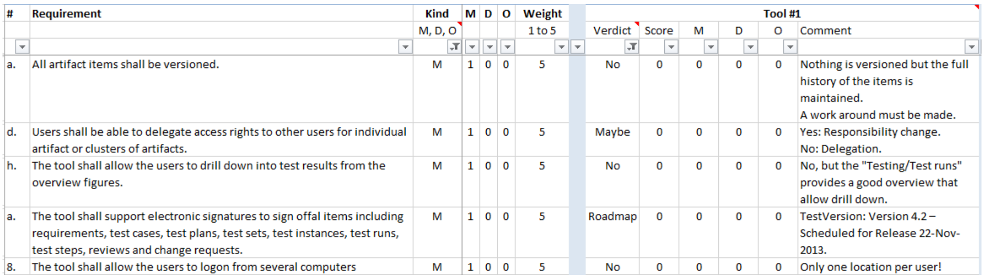
The tool was installed in a protected environment, configured and customized to reflect the work processes of the company. However, some processes had to be modified to reflect the change from document-centric to database-centric methods of writing requirements and test cases, perform testing, report defects, conduct reviews, etc.

Table 2.4: Requirements conformance for tool #1 (excerpt). M = Mandatory, D = Desired, O = Optional. The verdict "yes" gives full score 1.0*weight, "partial" gives half score 0.5*weight, and other verdicts gave zero score.



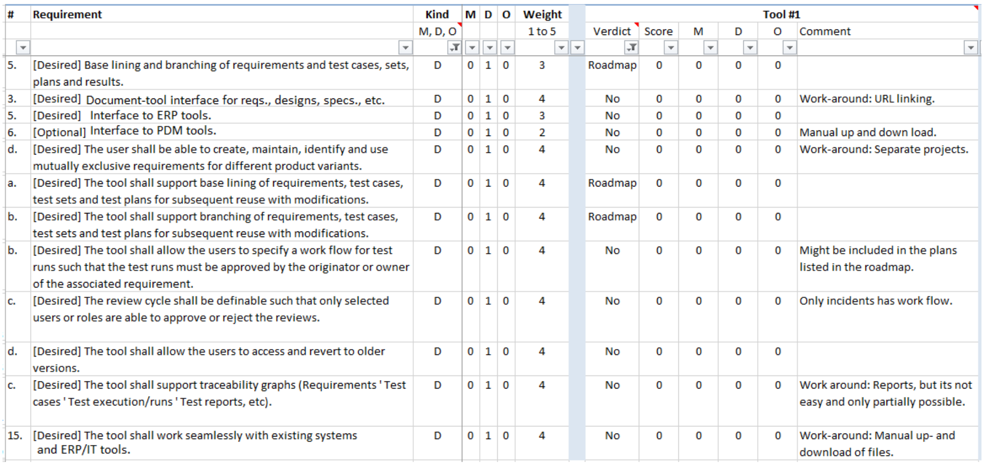
#	Requirement	Kind	M	D	O	Weight	Tool #1					
		M, D, O				1 to 5	Verdict	Score	M	D	O	Comment
1.	**Requirements management:**		**0**	**0**	**0**			0	0	0	0	
a.	The tool shall allow the users to create and maintain hierarchies or requirements for portfolios, product series, individual products, platforms, components, subcomponents, etc. The tool shall be able to handle requirements that apply to a single project/product as well as requirements that apply to many projects/products.	M	1	0	0	5	Yes	5	1	0	0	
b.	The tool shall support arbitrary linking between requirements across	M	1	0	0	5	Yes	5	1	0	0	
c.	The tool shall allow the users to create and maintain requirements of various definable kinds such as stakeholder wishes, mandatory requirements, optional requirements, hardware requirements, software requirements, design items (see point 2 below), etc. The user shall therefore be able to define any kind of requirement/item.	M	1	0	0	5	Yes	5	1	0	0	
d.	[Desired] The user shall be able to create, maintain, identify and use mutually exclusive requirements for different product variants.	D	0	1	0	4	No	0	0	0	0	Work-around: Separate projects.
e.	[Optional] The tool shall be able to analyze the quality and consistency of the requirements by means of semantic analysis or other relevant methodology. Ambiguous terms shall be highlighted.	O	0	0	1	3	No	0	0	0	0	
f.	The tool must be able to group requirements into clusters that correspond to requirement documents. Each group/cluster/document shall include an attribute for a textual description. The tool shall be able to generate a requirements document for each group or cluster of	M	1	0	0	5	Yes	5	1	0	0	Categories and hierarchies.
g.	The tool shall allow the users to define mandatory and optional attributes.	M	1	0	0	5	Yes	5	1	0	0	But default value seems to be necessary in order to be able to create the item.
h.	[Optional] The tool shall allow the users to add tags to requirements and test cases for subsequent search and filtering.	O	0	0	1	3	Partial	1,5	0	0	0,5	As custom fields.
i.	[Desired] The tool shall allow the users to add drawings and formulas to requirements descriptions	D	0	1	0	4	Partial	2	0	0,5	0	The user can add a picture by entering a URL to a file (must be
13.	**Traceability and analysis:**		**0**	**0**	**0**			0	0	0	0	
a.	The tool shall support electronic signatures to sign offal items including requirements, test cases, test plans, test sets, test instances, test runs, test steps, reviews and change requests.	M	1	0	0	5	Roadmap	0	0	0	0	TestVersion: Version 4.2 – Scheduled for Release 22-Nov-2013.
b.	The tool shall support easy linking of relationships by drag and drop, by drawing connections between items or by searching for relevant items.	M	1	0	0	5	Yes	5	1	0	0	Search plus bulk creation of e.g. test cases from requirements.
c.	[Desired] The tool shall support traceability graphs (Requirements ' Test cases ' Test execution/runs ' Test reports, etc).	D	0	1	0	4	No	0	0	0	0	Work around: Reports, but its not easy and only partially possible.
d.	**The tool shall be able to generate audit reports for US-FDA and other stakeholders.**	**M**	**1**	**0**	**0**	5	Yes	5	1	0	0	To be cheked!!!!
e.	[Desired] The tool shall allow the user to identify which versions/releases contain a known issue, bug or non-conformance to test acceptance criteria.	D	0	1	0	4	Partial	2	0	0,5	0	Detected and solved by default. Other releases can be described in attributes.
f.	The tool shall allow the users to trace linked item in both up and down	M	1	0	0	5	Yes	5	1	0	0	
g.	[Optional] The tool may support traceability into attached documents.	O	0	0	1	3	No	0	0	0	0	
h.	[Optional] The tool may support traceability into source code and other design items.	O	0	0	1	3	Partial	1,5	0	0	0,5	Design items based on the requirement item: Yes. Otherwise: No.
i.	The tool shall be able to analyze the impact of changes to requirements or test cases such that all linked items are highlighted. Subscribers and owners of affected items shall be notified accordingly.	M	1	0	0	5	Partial	2,5	0,5	0	0	Change notification to subscribers and default users for changed items but not linked items.
15.	[Desired] The tool shall work seamlessly with existing systems and ERP/IT tools.	D	0	1	0	4	No	0	0	0	0	Work-around: Manual up- and download of files.
16.	The tool shall provide easy search, browse and filtering based on tags, attributes and textual descriptions, user names, etc.	D	0	1	0	5	Yes	5	0	1	0	
17.	[Desired] The tool shall be able to show traceability between items graphically as a tree or as a graph.	D	0	1	0	4	Partial	2	0	0,5	0	Only trees of similar itemtypes OR as reports of mixed items.
18.	The tool shall have fast response times such that more than 90% of all operations are completed in less than 1 second; and no operation takes more than 5 seconds to complete.	M	1	0	0	5	Yes	5	1	0	0	To be verified (evaluation on hosted service in USA is somewhat slower).
Conclusion			0	0	0				0	0	0	
1.	**Total score**		81	60	42	717	NA	503	65	39,5	19,5	The results are entirely based on my own evaluations.
2.	**Total score in percentage of obtainable score**					**100,0%**		**70,2%**	**80,2%**	**65,8%**	**46,4%**	
	Gap = Specification - Conformance =							**29,8%**	**19,8%**	**34,2%**	**53,6%**	

Table 2.5: Requirements gap for tool #1 (mandatory requirements only).



#	Requirement	Kind	M	D	O	Weight	Tool #1					
		M, D, O				1 to 5	Verdict	Score	M	D	O	Comment
a.	All artifact items shall be versioned.	M	1	0	0	5	No	0	0	0	0	Nothing is versioned but the full history of the items is maintained. A work around must be made.
d.	Users shall be able to delegate access rights to other users for individual artifact or clusters of artifacts.	M	1	0	0	5	Maybe	0	0	0	0	Yes: Responsibility change. No: Delegation.
h.	The tool shall allow the users to drill down into test results from the overview figures.	M	1	0	0	5	No	0	0	0	0	No, but the "Testing/Test runs" provides a good overview that allow drill down.
a.	The tool shall support electronic signatures to sign offal items including requirements, test cases, test plans, test sets, test instances, test runs, test steps, reviews and change requests.	M	1	0	0	5	Roadmap	0	0	0	0	TestVersion: Version 4.2 – Scheduled for Release 22-Nov-2013.
8.	The tool shall allow the users to logon from several computers	M	1	0	0	5	No	0	0	0	0	Only one location per user!

Table 2.6: Requirements gap for tool #1 (desired requirements only).



#	Requirement	Kind	M	D	O	Weight	Tool #1					
		M, D, O				1 to 5	Verdict	Score	M	D	O	Comment
5.	[Desired] Base lining and branching of requirements and test cases, sets, plans and results.	D	0	1	0	3	Roadmap	0	0	0	0	
3.	[Desired] Document-tool interface for reqs., designs, specs., etc.	D	0	1	0	4	No	0	0	0	0	Work-around: URL linking.
5.	[Desired] Interface to ERP tools.	D	0	1	0	3	No	0	0	0	0	
6.	[Optional] Interface to PDM tools.	D	0	1	0	2	No	0	0	0	0	Manual up and down load.
d.	[Desired] The user shall be able to create, maintain, identify and use mutually exclusive requirements for different product variants.	D	0	1	0	4	No	0	0	0	0	Work-around: Separate projects.
a.	[Desired] The tool shall support base lining of requirements, test cases, test sets and test plans for subsequent reuse with modifications.	D	0	1	0	4	Roadmap	0	0	0	0	
b.	[Desired] The tool shall support branching of requirements, test cases, test sets and test plans for subsequent reuse with modifications.	D	0	1	0	4	Roadmap	0	0	0	0	
b.	[Desired] The tool shall allow the users to specify a work flow for test runs such that the test runs must be approved by the originator or owner of the associated requirement.	D	0	1	0	4	No	0	0	0	0	Might be included in the plans listed in the roadmap.
c.	[Desired] The review cycle shall be definable such that only selected users or roles are able to approve or reject the reviews.	D	0	1	0	4	No	0	0	0	0	Only incidents has work flow.
d.	[Desired] The tool shall allow the users to access and revert to older versions.	D	0	1	0	4	No	0	0	0	0	
c.	[Desired] The tool shall support traceability graphs (Requirements ' Test cases ' Test execution/runs ' Test reports, etc).	D	0	1	0	4	No	0	0	0	0	Work around: Reports, but its not easy and only partially possible.
15.	[Desired] The tool shall work seamlessly with existing systems and ERP/IT tools.	D	0	1	0	4	No	0	0	0	0	Work-around: Manual up- and download of files.

User guides were created and training of developers, testers, reviewers, managers, etc., was commenced. Existing requirements and test cases were migrated to the tool for a pilot project that lasted 3 months. The tool was validated according to internal company standards 1 month after the pilot project.

One year after the implementation started, the tool was used for all development projects by 400 users from 5 different sites distributed around the globe. The project entered maintenance mode 3 month later. The tool has been in continual operation since 2013, with only minor changes to the configurations and setup.

The tool has successfully achieved the primary objective (automatic generation of traceability reports for various authorities), secondary objectives (improved management of requirements, testing, defects, etc.), and some unplanned positive side effects (clarification and improvement of work processes).

2.7 Case study: Modeling tools (multi-tool scenario)

The proposed method was applied to selecting modeling tools for pilot modeling cases at GN Hearing. The tools should enable several modeling disciplines. It was decided early to focus on multi-tool scenarios because no single tool was expected to provide a satisfying solution. The steps of the method were applied in the following manner:

Step 1 - Justify and initiate change project:

The hearing systems from GN Hearing are becoming increasingly complex and it is required to reduce the risks of making errors in the specification and development phases. Errors in the requirements specification are particularly problematic because such errors typically are found very late. These observations follow the general trend for embedded systems as described by Ebert and Jones (2009) [3]. Many believe that MBSE can address these and other problems. Friedenthal et al. (2012) [76, p.20] outlines the following potential benefits of MBSE: Enhanced communication, reduced risks, improved quality, increased productivity, etc. GN Hearing has therefore initiated activities to determine if MBSE shall be included in future SE work.

Introduction of MBSE to the company was considered a large step even though modeling is used in various sub-disciplines. It was therefore decided to run a series of pilot projects to evaluate if and how to introduce systems modeling.

Step 2 - Develop specification and evaluation criteria:

From knowledge about the organization and its environment, a list of relevant stakeholder were identified and prioritized. The systems engineering team is the primary stakeholder. Secondary stakeholders are management, testers, software and hardware developers, audiologists, marketing, production, and a few others.

Informal interviews with stakeholders revealed huge differences in their perception and expectations of MBSE. Presentation of various "definitions" of MBSE did not streamline the perceptions. It was clear that a complete categorization of the various sub-disciplines of MBSE was needed to ask questions that are more precise. We therefore introduced a classification of MBSE as described in section 2.5.

From the knowledge about the types of systems under consideration (section 1.4 and 2.4) and the characteristics of the modeling disciplines (section 2.5), we prepared questionnaires for subsequent interviews. First, we asked which of the modeling disciplines were relevant for the organization. Next, we asked questions that were more detailed. As an example, we asked if the organization should be able to model e.g. state machines, activities, timing constraints as part of the fundamental modeling. For the behavioral simulation modeling, we asked if the organization should be able to model and simulate homogeneous or heterogeneous systems; etc. General questions concerning trustworthiness of vendors and tools, affordability, availability, etc., were also considered as recommended by Jadhav and Sonar [33], Nikoukaran et al. [24], and others.

After conducting the interviews, the answers were analyzed presented to relevant stakeholders. The analysis showed that most parts of fundamental modeling, behavioral simulation modeling, architectural analyses modeling, architecture/behavior co-modeling, and coherent modeling were needed. It was decided to postpone the decision about integrated engineering until the more basic modeling disciplines are mastered. Enterprise modeling is outside the scope of the company and it is therefore not included in the selection criteria. The output of the process was a table of selection criteria for the various modeling sub-disciplines. The selection criteria were prioritized and expressed in decreasing order of importance as "yes", "maybe", "later" or "no", as shown in table 2.7. The criteria may be refined further into an actual specification of the required technology.

Step 3 - Investigate tools and select best candidates:

Potential tool candidates were identified from searching the internet intensively. Rashidi (2016) [31] provided a list of 62 simulation tools. Tools for the systems or unified modeling languages (SysML, UML) were identified from websites such as

Weilkiens [38], Diagramming.org [43], Softdevtools.com [44], Modeling-languages.com [45], and from other sources. For GN Hearing, the tools listed in section 2.1 were identified as potential technologies (methodologies, languages, tools).

After installing the tools, tutorials were executed to learn how to use the tools, and to find out their capabilities. The rating of the technologies was based on comprehensive collection of information and extensive experimentation with different tools on smaller pilot projects. A few criteria were assessed by other means, e.g. by information obtained from others, and some criteria for some tools could unfortunately not be assessed by testing due to bugs, missing functionality, wrong configurations of demo tools, etc. We used a numbering scheme with ratings from 1 to 5 stars representing the degree of conformance to the criteria. The ratings have different meanings for different criteria and different technologies. Some criteria could be analyzed precisely, where others were more difficult to evaluate (e.g. conformance to usability requirements). The confidence or precision of the ratings therefore vary. Stars are thus used instead of numerical values to avoid giving an impression of a high degree of precision. However, the star ratings do show the feasibility of each technology with regard to different criteria.

Table 2.7 shows an extract of the evaluation results for some of the most relevant tools. Numerous other tools and technologies have been evaluated but not shown in the table because they were infeasible or otherwise unattractive. Some of the other relevant technologies are mentioned in the modeling guides in chapter 2 in [1], though. The amount of work required to evaluate the tools was not recorded but took several weeks.

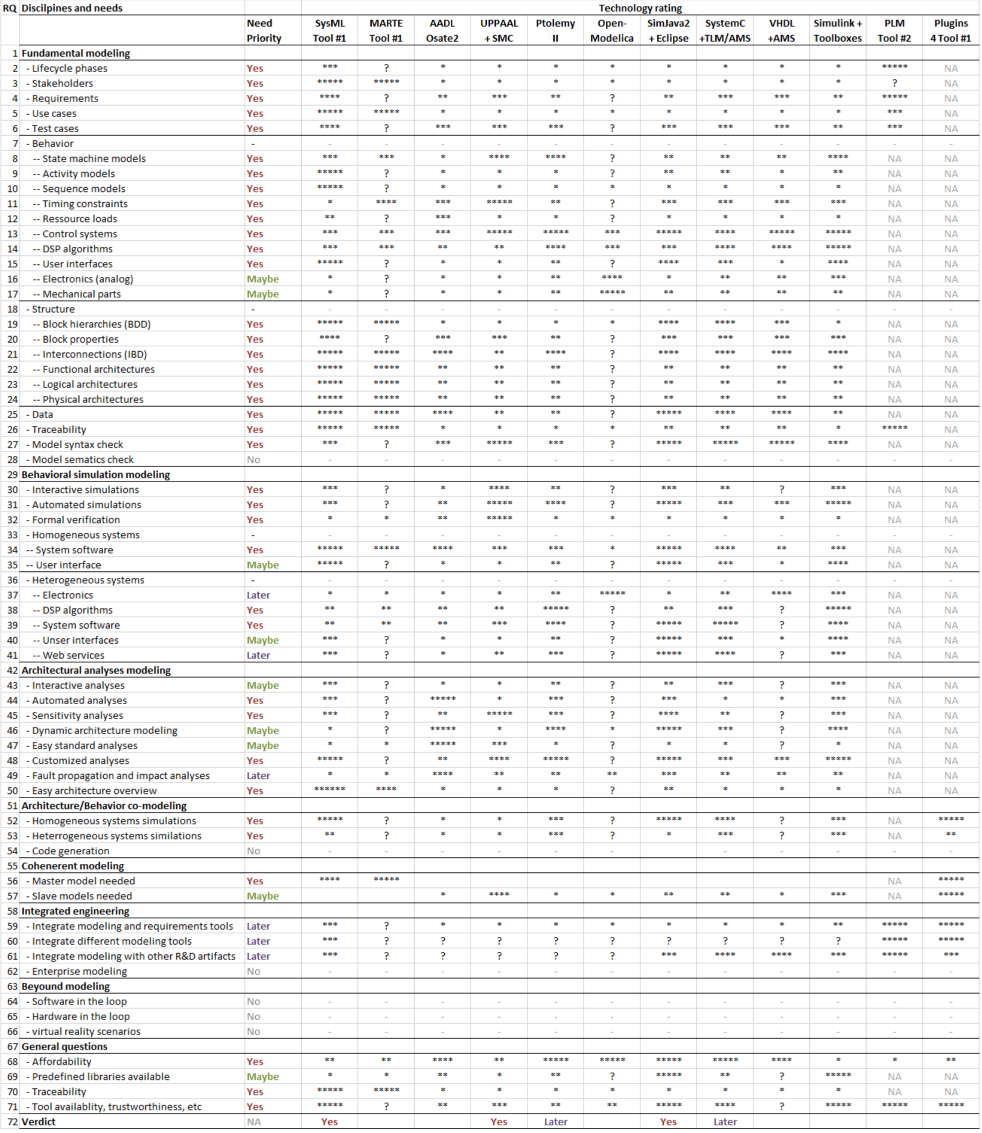
For this case study, the multi-tool selection method described in section 2.3.2 was used because (1) no single-tool solution was close to fulfilling all evaluation criteria, and (2) using separate tools for different modeling disciplines was found acceptable. The analysis is presented in chapter 2.2 in [2]. From the analysis, the following set of tools were identified (RQ72r):

```
{ {SysML and AADL and UPPAAL and OpenModelica and PLM} and
  {PtolemyII or SystemC} and {SimJava2 or Simulink} }
```

The AADL tool could be ignored because SysML cover the most essential needs for AAM. The OpenModelica could be ignored because the requirements were not mandatory. PLM tool plugins could be ignored because CM and IE is beyond the scope of the company's current modeling plans. Simulink could be excluded because PtolemyII + SimJava2 + SystemsC are more affordable and provide similar functionality. The final setup could therefore be reduced as described in the appendix (RQ72f):

```
{ {SysML and UPPAAL and SimJava2} and {PtolemyII or SystemC} }
```

Table 2.7: Example of table showing the correlation between modeling needs and features provided by different technologies. Notice that the different technologies are not necessarily comparable because they address vastly different needs using notably different means. The table is further discussed in section 2.2 in [2]. The table is enlarged as table 2.1, 2.2 and 2.3 in [2] for improved readability.



RQ	Discilpines and needs		Technology rating											
		Need Priority	SysML Tool #1	MARTE Tool #1	AADL Osate2	UPPAAL + SMC	Ptolemy II	Open-Modelica	SimJava2 + Eclipse	SystemC +TLM/AMS	VHDL +AMS	Simulink + Toolboxes	PLM Tool #2	Plugins 4 Tool #1
1	**Fundamental modeling**													
2	- Lifecycle phases	Yes	***	?	*	*	*	*	*	*	*	*	*****	NA
3	- Stakeholders	Yes	*****	*****	*	*	*	*	*	*	*	*	?	NA
4	- Requirements	Yes	****	?	**	***	**	?	**	***	***	**	*****	NA
5	- Use cases	Yes	*****	*****	*	*	*	*	*	*	*	*	***	NA
6	- Test cases	Yes	****	?	***	***	***	?	***	***	***	**	***	NA
7	- Behavior	-	-	-	-	-	-	-	-	-	-	-	-	-
8	-- State machine models	Yes	***	***	*	****	****	?	**	**	**	****	NA	NA
9	-- Activity models	Yes	*****	?	*	*	*	?	**	**	*	**	NA	NA
10	-- Sequence models	Yes	*****	?	*	*	*	*	*	*	*	*	NA	NA
11	-- Timing constraints	Yes	*	****	***	*****	**	?	***	***	***	***	NA	NA
12	-- Ressource loads	Yes	**	?	***	*	*	?	*	*	*	*	NA	NA
13	-- Control systems	Yes	***	***	***	*****	*****	***	*****	****	*****	*****	NA	NA
14	-- DSP algorithms	Yes	***	***	**	**	****	***	***	****	****	*****	NA	NA
15	-- User interfaces	Yes	*****	?	*	*	**	?	****	***	*	****	NA	NA
16	-- Electronics (analog)	Maybe	*	?	*	*	**	****	*	**	**	***	NA	NA
17	-- Mechanical parts	Maybe	*	?	*	*	**	*****	**	**	**	**	NA	NA
18	- Structure	-	-	-	-	-	-	-	-	-	-	-	-	-
19	-- Block hierarchies (BDD)	Yes	*****	*****	*	*	*	*	****	****	***	*	NA	NA
20	-- Block properties	Yes	****	?	***	***	**	?	***	***	***	***	NA	NA
21	-- Interconnections (IBD)	Yes	*****	*****	****	**	****	?	****	****	****	****	NA	NA
22	-- Functional architectures	Yes	*****	*****	**	**	**	?	**	**	**	**	NA	NA
23	-- Logical architectures	Yes	*****	*****	**	**	**	?	**	**	**	**	NA	NA
24	-- Physical architectures	Yes	*****	*****	**	**	**	?	**	**	**	**	NA	NA
25	- Data	Yes	*****	*****	****	**	**	?	*****	****	****	**	NA	NA
26	- Traceability	Yes	*****	*****	*	*	*	*	**	**	**	*	*****	NA
27	- Model syntax check	Yes	***	?	***	*****	***	?	*****	*****	*****	****	NA	NA
28	- Model sematics check	No	-	-	-	-	-	-	-	-	-	-	-	-
29	**Behavioral simulation modeling**													
30	- Interactive simulations	Yes	***	?	*	****	**	?	***	**	?	***	NA	NA
31	- Automated simulations	Yes	***	?	**	*****	****	?	*****	***	***	*****	NA	NA
32	- Formal verification	Yes	*	*	**	*****	*	*	*	*	*	*	NA	NA
33	- Homogeneous systems	-	-	-	-	-	-	-	-	-	-	-	-	-
34	-- System software	Yes	*****	*****	****	***	***	*	*****	****	**	***	NA	NA
35	-- User interface	Maybe	*****	?	*	*	**	?	*****	***	*	****	NA	NA
36	- Heterogeneous systems	-	-	-	-	-	-	-	-	-	-	-	-	-
37	-- Electronics	Later	*	*	*	*	**	*****	*	**	****	***	NA	NA
38	-- DSP algorithms	Yes	**	**	**	**	*****	?	**	***	?	*****	NA	NA
39	-- System software	Yes	**	**	**	***	****	?	*****	*****	?	****	NA	NA
40	-- Unser interfaces	Maybe	***	?	*	*	**	?	*****	***	*	****	NA	NA
41	-- Web services	Later	***	?	*	**	***	?	*****	****	?	***	NA	NA
42	**Architectural analyses modeling**													
43	- Interactive analyses	Maybe	***	?	*	*	**	?	**	***	?	***	NA	NA
44	- Automated analyses	Yes	***	?	*****	*	***	?	***	*	*	***	NA	NA
45	- Sensitivity analyses	Yes	***	?	**	*****	***	?	****	**	?	***	NA	NA
46	- Dynamic architecture modeling	Maybe	*	?	*****	*	****	*	*****	***	?	****	NA	NA
47	- Easy standard analyses	Maybe	*	*	*****	***	*	?	*	*	?	*	NA	NA
48	- Customized analyses	Yes	*****	?	**	****	*****	?	*****	***	***	*****	NA	NA
49	- Fault propagation and impact analyses	Later	*	*	****	**	**	**	***	**	**	**	NA	NA
50	- Easy architecture overview	Yes	******	****	*	*	*	?	**	*	*	*	NA	NA
51	**Architecture/Behavior co-modeling**													
52	- Homogeneous systems simulations	Yes	*****	?	*	*	***	?	*****	****	?	***	NA	*****
53	- Heterrogeneous systems similations	Yes	**	?	*	*	***	?	*	***	?	***	NA	**
54	- Code generation	No	-	-	-	-	-	-		-	-	-	-	-
55	**Cohenerent modeling**													
56	- Master model needed	Yes	****	*****									NA	*****
57	- Slave models needed	Maybe			*	****	*	*	**	**	*	***	NA	*****
58	**Integrated engineering**													
59	- Integrate modeling and requirements tools	Later	***	?	*	*	*	*	*	*	*	**	*****	*****
60	- Integrate different modeling tools	Later	***	?	?	?	?	?	?	?	?	?	*****	*****
61	- Integrate modeling with other R&D artifacts	Later	***	?	?	?	?	?	***	****	****	***	*****	***
62	- Enterprise modeling	No	-	-	-	-	-	-	-	-	-	-	-	-
63	**Beyound modeling**													
64	- Software in the loop	No	-	-	-	-	-	-	-	-	-	-	-	-
65	- Hardware in the loop	No	-	-	-	-	-	-	-	-	-	-	-	-
66	- virtual reality scenarios	No	-	-	-	-	-	-	-	-	-	-	-	-
67	**General questions**													
68	- Affordability	Yes	**	**	****	**	*****	*****	*****	*****	****	*	*	**
69	- Predefined libraries available	Maybe	*	*	**	*	**	?	*****	**	?	*****	NA	NA
70	- Traceability	Yes	*****	*****	*	*	*	*	*	*	*	*	NA	NA
71	- Tool availablity, trustworthiness, etc	Yes	*****	?	**	***	**	**	*****	****	?	*****	*****	*****
72	**Verdict**	NA	Yes			Yes	Later		Yes	Later				

PtolemyII and SystemC-TLM/AMS were included mainly for modeling heterogeneous systems. However, the need for modeling such systems is not imminent. The use of PtolemyII and SystemC can therefore be postponed. The set of tools for the pilot cases and initial setup could therefore be reduced as described in the appendix (RQ72i):

```
{SysML and UPPAAL and SimJava2}
```

SysML will mainly be used for fundamental modeling. Examples of SysML models are included in chapter 2 in [1]. UPPAAL will primarily be used for test-driven modeling of embedded systems as described in chapter 3. SimJava2 will mainly be used for test-driven modeling of smart systems as described in chapter 4.

Step 4 - Implement, maintain, and evolve solution:

The company has purchased licenses for a SysML tool, which has been installed and used for creating diagrams that were included in architecture description documents. Furthermore, the SysML tool and an evaluation version of UPPAAL (including SMC) have been used on a pilot case for test-driven modeling of embedded systems as described in section 3.6. The combination of SysML, UPPAAL and SimJava2 has been used on a pilot case for modeling a large and complex smart system as described in section 4.8. Ptolemy-II has been used during the evaluation period to create and simulate small models of elements of hearing systems. The remaining modeling technologies have also been used during the evaluation period to create various systems models, meta models, model transformations, special menus and tool plugins for demonstrating the feasibility of potential future upgrades to the chosen modeling setup.

Redesign of development process, making of user guides, conducting of training, migrating of data, and reorganizing the company are postponed until the full learning of pilot cases are analyzed and incorporated into an overall MBSE strategy.

The complexity of embedded systems is growing almost exponentially, according to Ebert (2009) [3]. It is therefore likely that the analysis has to be repeated in the future to maintain and evolve the tool set such that modeling scenarios of the future can be handled. A need for simulators that are faster and can handle extremely large models is already apparent.

2.8 Discussion

2.8.1 Results

The proposed method was applied to the selection of a traceability tool for the medical device company GN Hearing as described in see section 2.6. A large collection of tools were investigated, tested, and evaluated. Tools that were ranked higher than the selected tool were dismissed because the gap analyses showed that they missed essential requirements. The selected tool has been in continual operation since 2013. It is used daily for tracing and managing requirements, implementations, tests, bugs, defects, issues, tasks, etc.

The proposed method was also used for identifying a set of modeling tools that meets the immediate and some future needs of GN Hearing.

SysML was identified as a good modeling language for FM. Licenses for a commercial tool have been purchased. The tool has been used in case studies as described in chapter 3 and 4.

UPPAAL was identified as good choice for formal modeling of some parts of BSM, AAM and ABCM. UPPAAL-SMC was similarly identified as a good choice for statistical model checking. Both standard UPPAAL and UPPAAL-SMC have been used in case studies as described in chapter 3 and 4.

SimJava2 was identified as a good choice for modeling and simulating systems that are too large or complex for UPPAAL. SimJava2 was used with the Eclipse tool to model and simulate a smart system in a case study as described in chapter 4.

Simulink was identified as a good choice for modeling and simulating heterogeneous system models. This tool can be used for both BSM, AAM and ABCM. Furthermore, it can be integrated with the chosen SysML modeling tool to facilitate complex parametric simulations. The feasibility of Simulink has been demonstrated on small modeling projects but a larger modeling case has not been conducted yet.

Other modeling technologies have also been used during the evaluation period to create various systems models, meta models, model transformations, special menus and tool plugins for demonstrating the feasibility of potential future upgrades to the chosen modeling setup. The more advanced forms of modeling are postponed until the setup with SysML, UPPAAL and SimJava2 has been proved useful on real projects in the company.

2.8.2 Observations

GN Hearing has a dedicated department for SE. This department is responsible for defining top-level requirements, defining and executing system tests, defining system architectures, and facilitating communication between different stakeholders. Modeling has only be used occasionally in the SE department.

However, various forms of modeling and simulating are used in other departments for software design, chip design, antenna design, mechanical design, acoustical design, etc. Each discipline uses different modeling languages and tools that cannot easily be integrated for holistic systems modeling.

The company has successfully developed and marketed a large range of product using the existing setup. Great success in the past could lead to lack of motivation to change the setup, because the benefits of implementing a new, costly setup might be perceived as unnecessary. However, the increasing complexity of hearing systems has demonstrated that new methods and processes are needed for development of future systems. Thus, there is a general acceptance in the company of investigating if modeling of systems, software, etc., should be part of future development methods.

2.8.3 Learning points

Early in this PhD project, it was suspected that single-tool solutions were unable to satisfy all modeling needs of the company. Different objectives require vastly different features of the modeling tools and languages as seen from the following few examples:

- Modeling for communication requires that the tools are capable of providing specialized views, reports or presentations for individual stakeholders.
- Modeling for verification of system functionality and performance requires that the tools are able to simulate or formally verify the models, or to generate simulation code that can be executed with external simulations.
- Modeling for traceability requires that the modeling languages include elements and notations for linking model items, such that views, reports or tables of relationships can be generated.

From a few informal interviews of people involved in systems engineering at various companies and other organizations, it was realized that solid methods for selecting tools were rarely used. Often, tools are selected without comprehensive analyses of actual needs and available tools. The proposed method was develop for allowing GN Hearing to conduct a more thorough analysis.

Vendors often promise more than the tool can actually do. Questions concerning unsupported features are often reformulated by vendors such that circumventions can be identified. This type of behavior is most praiseworthy, because it allows the potential users to find new and better ways of using the tools. However, blind trust in the statements of vendors may lead to selecting tools that are not optimal for the organization. Tools may safely be *disqualified* by assessing vendors' information, but candidates should not be *qualified* without further evaluation, where the tools are tested by creating models that reflect the intended use.

Modeling is generally difficult and most disciplines have very steep learning curves. Learning how to model certain system aspects in various languages can be very time consuming because modeling communities seldom offer more than rudimentary help. Modeling tools are often filled with many features that require much time and effort to learn to use correctly and optimally. Thus, the evaluation of tools can be very time consuming, and there may be a risk of wrong conclusions because the language constructs or features are used and assessed wrongly. However, many tool vendors are willing to assist greatly during tool evaluation periods.

2.8.4 Advantages

The proposed method (section 2.3) has the following advantages over previous methods for selecting modeling tools:

- The method is systematic and includes steps for justifying and initiating the tool change project, developing specification and evaluation criteria, investigating tools and selecting candidates, implementing, maintaining and evolving the chosen solution.
- The method is practical and does not require expensive special tools for calculating evaluation scores and tool rankings.
- The method for specifying the tool requirements includes thorough analysis of the modeling disciplines of interest for the company. This makes it easy to include future modeling scenarios that are otherwise easily overlooked.
- The method can be applied to both single-tool and multi-tool scenarios. Single-tool solutions are applicable for many SE disciplines, e.g. requirements management. However, multi-tool solutions are often required for MBSE, because there is no current single-tool solution that covers all modeling needs.

- The method for selecting multi-tool solutions is based on set theory and simple operations that allow companies to identify feasible sets of tools that cover all essential needs.
- The method for selecting multi-tool solutions uses ranking of the identified sets of tools to find the best set. The proposed method therefore has a high likelihood of identifying the (sets of) tools that best meet the actual needs of the company.
- The method for selecting tools include gap analyses to validate the feasibility of the chosen tool set. The proposed method therefore has a low risk of selecting tools that fail to meet essential requirements.

Some of the previous methods are also systematic and practical, but do not have any of the other advantages. No previous work considers multi-tool scenarios. Ranking methods are also used in previous work, but only for individual tools and not for set of tools. No previous work includes gap analysis to validate the chosen tools.

2.8.5 Disadvantages

The proposed method has the following disadvantages compared to previous methods for selecting modeling tools:

- Using the proposed method is time consuming and many resources are required to conduct all included steps systematically. Highlighting all steps before commencing the analyses may cause decision paralysis, which may lead to using less rigorous selection methods or to abandoning the tool selection altogether.
- Knowledge of various modeling disciplines is needed to develop the tool specification and evaluations criteria successfully. Identifying analysts with deep knowledge of all relevant modeling disciplines may be difficult. Using people with limited skills to perform the analyses introduces risks of overlooking essential needs.
- Detailed analyses of all required features for every tool candidate can be very time-consuming and may require modeling skills that have not been acquired in advance. This may cause incomplete or erroneous evaluations, which in turn may lead to selecting infeasible tools.

These potential disadvantages should be mitigated to benefit from the advantages of the proposed method.

2.8.6 Limitations

The proposed method is intended for selection of tools for various MBSE disciplines. However, the method is generally applicable to selection of SE tools, as discussed in [73].

The method has been applied to selection of a traceability tool (appendix 2.6) and a set of modeling tools (section 2.7) for GN Hearing. The traceability tool has been in successful operation since 2013. The set of modeling tools has only been used on pilot projects. The validation of the method is therefore limited for multi-tool scenarios.

2.8.7 Improvements

Needs for improving the proposed method have not been identified so far. However, some steps of the method could benefit from automating, particularly for multi-tool scenarios.

In the modeling tool selection case (section 2.7), a spreadsheet was used for the correlation matrix (table 2.7) to give an overview of tool evaluations. Composing the set of tools for each requirement requires that properties of different tools be compared even though they may not be directly comparable. This part of the process may be affected by irrational decisions, if the analyst has bias towards some of the evaluated tools. Manually performing the operations to obtain the feasible sets that cover all requirements is probably error-prone as well. The usability of the method could therefore be improved by developing a tool to analyze the feasible sets of tools by using the equations (2.2) to (2.5).

The ranking of tools for single-tool scenarios is easily performed by using equation (2.1). This can easily be done in spreadsheets as shown in the example in section 2.6. However, ranking of individual tools is irrelevant for multi-tool scenarios because it is the rank of the entire set of tools, which is important. Using spreadsheets for ranking set of tools is cumbersome. The usability of the method could therefore be further improved by developing a tool to calculate the rank of all feasible sets of tools.

Analyzing the gaps between tool capabilities and the required, desired and optional requirements is easily done by using spreadsheet tools with filtering functions for single-tool scenarios. However, using this technique is cumbersome for multi-tool scenarios. The usability of the method could therefore be even further improved by developing a tool to analyze the gaps for all feasible sets of tools.

2.8.8 Future work

From the description of potential improvements (section 2.8.7), it can be deduced that development of analysis tools for multi-tool selection scenarios is among the candidates for future work. Even though it is recommended to perform the analyses at regular intervals, it will nevertheless be done with a low frequency. The benefits of having good analysis tools therefore cannot justify the effort required to develop and maintain such tools for most separate organizations. However, modeling communities may be interested in future work on commercial or open source tools that can be easily be used by analysts from many different companies.

The modeling tools that were selected in the case study (section 2.7) have only been applied to pilot projects. Implementing the tools setup for daily use is the most obvious future work that must be conducted in the company.

Incorporating SysML for FM has been initiated and licenses have been purchased. To leverage SysML fully, additional future work has to be commenced. It must be decided how FM shall be included in the systems engineering processes. Currently, requirements are managed in document for the early phases and in a traceability tool for later phases. The current version of the SysML tool cannot replace the existing traceability tool because SysML lacks essential features (such as version control, access control, automatic history logging, etc.) for managing requirements, tests, defects, tasks, etc. However, SysML provides a holistic approach to requirements engineering and system design that cannot be obtained with the previously used tools. Integrating FM and SysML into the existing processes and tool setup is a major issue that the company should address in the future work. Training of systems engineers and domain experts in the FM discipline, the SysML formalism, and the chosen SysML tool should also be undertaken before SysML is launched for daily use.

Other engineering disciplines such as software development and chip design may also benefit from parts of FM. The future work should include an analysis of how to integrate FM for systems engineering with these other disciplines. Especially it should be determined if SysML can be used as a common modeling language or if the other engineering disciplines have requirements that call for different languages. If different languages are required, the question of integration emerges, which should be further investigated.

Other activities in the company (e.g. marketing) may similarly benefit from adopting FM with SysML for modeling of scenarios, etc. The future work should include an analysis of the benefit of expanding the use of FM with SysML or similar language to the entire organization.

The case studies that are described in chapter 3 and 4 are examples of modeling with elements of BSM, AAM and ABCM. This form of modeling can be introduced independently of FM. The future work should identify cases where these forms of modeling can be beneficial. Training in the formalism of UPPAAL and SimJava2 (or equivalent languages) is required before they can be launched for daily use.

The CM, IE and EM modeling disciplines are currently out of scope because they do not add value unless FM, BSM, AAM or ABCM are practiced on a daily basis. However, the company may prepare their introduction as part of the future work.

The application of the proposed method should be repeated at regular intervals because new needs and new technologies emerge over time. Rather than leaving it to chance, it may be beneficial to define a fixed interval at which the analyses should be repeated, e.g. every third year.

The application of the proposed method was limited to two cases as described in section 2.7 and 2.6. The case of selecting a traceability tool verifies all steps of the proposed method for single-tool scenarios. The additional steps for multi-tool scenarios were applied to the case of selecting modeling tools. However, the modeling tools have not yet been implemented for daily use, which prevents exhaustive verification. The future work should therefore include a more thorough validation of the proposed method, especially for multi-tool scenarios.

2.8.9 Conclusion

This chapter has addressed the need for a systematic and practical method for selecting MBSE tools. Reviewing literature of related work showed that:

(1) Comprehensive methods for selection of MBSE tools have not been reported. Some authors provided generic advices. Other authors compared specific tools. Most selection methods concern other domains than MBSE, such as selection of vendors, RM tools, simulation tools, etc.

(2) There is no common understanding of what MBSE includes and which features MBSE tools must provide.

(3) No reported method considers the case where multi-tool solutions are needed to meet all essential needs. Reported methods only consider the single-tool case, where the best possible tool is selected and its imperfections are accepted.

(4) Reported methods use either simple qualitative methods (e.g. feature evaluation) or quantitative methods (e.g. WSM, AHP) for ranking tools.

(5) No reported method specifically addresses the problem of verification and validation, i.e. analyzing if the selected tool solutions are acceptable and meet all essential requirements.

(6) Most reported work appears to be of academic origin without any industrial validation.

The proposed method was developed to overcome limitations of previous methods and was specifically developed for the domain of MBSE, but it can be used in generalized form for selecting other types of SE tools. The method was developed from existing methods by combining elements from various works and by adding new elements that are unique for this approach. It describes how to

1. Justify and initiate projects for selecting or changing tools.
2. Develop tool specifications and evaluation criteria.
3. Investigate and select best tool candidates.
4. Implement, maintain and evolve the chosen solution.

Proper use of the proposed method require

1. Knowledge about various characteristics of the systems under consideration because they pose different modeling needs.
2. Knowledge about modeling disciplines because they address different needs and objectives.

The systems under consideration are classified according to business model, value proposition, quality requirements, usage scenarios, composition (simple, complex, homogeneous, heterogeneous), major modeling and simulating criteria, constituent hardware and software, etc. The modeling and related disciplines are classified into fundamental modeling, behavioral simulating modeling, architectural analyses modeling, architecture/behavior co-modeling, coherent modeling, enterprise modeling, other engineering activities, and activities beyond engineering. The classifications of systems and disciplines are used for determining the tool requirements. These requirements are subsequently translated into prioritized evaluation criteria.

Each evaluation criteria is investigated for each modeling technology (method, language and tool). Tables that correlate needs and capability are thus created during the application of the method. For single-tool solutions, the candidates are ranked by

WSM. For multi-tool solutions, feasible sets of tools that meet all requirements are obtained by combining the sets that are needed to meet each individual requirement. The combined sets are subsequently ranked by WSM. Analyses of the gaps between needs and capabilities are finally performed to identify the most acceptable choice for both single- and multi-tool scenarios.

The proposed method has been applied for selecting a traceability tool for the medical device company GN Hearing. A large collection of tools were investigated, tested, and evaluated. Tools that were ranked higher than the selected tool were dismissed because the gap analyses showed that they missed essential requirements. The selected tool has been in continual operation since 2013. It is used for tracing and managing requirements, implementations, tests, bugs, defects, issues, tasks, etc.

The proposed method has also been applied to selecting a set of modeling and simulation tools for SE at GN Hearing. A large collection of modeling technologies were investigated, tested and evaluated. A set of 3 tools for the initial setup was identified. The chosen setup may be extended by other tools that were identified. The tools in the initial setup has been used in the case studies in chapter 3 and 4. Other tools that were identified for possible future extensions have been used on evaluation projects.

Proper verification of the proposed method and selection techniques requires more than two cases so the future work should concentrate on identifying and executing more cases to get statistical significance. However, it is hoped that the proposals presented in this chapter may help others in the process of selecting SE tools henceforth.

2.9 References

[1] Allan Munck
"Model-Based Systems Engineering Guidelines."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[2] Allan Munck
"Modeling tool selection."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[3] Christof Ebert and Casper Jones.
"Embedded software: Facts, figures and future."
IEEE Computer Magazine, 2009, Volume 42, Issue 4.

[4] Confederation of Danish Industry (DI), Community of IT, Tele, Electronics and Communication (ITEK), Project Industrial Technology and Software (ITOS).

Website (last visited 2014.04.24):
`http://itek.di.dk/projekter/itos/`

[5] INCOSE SE Handbook Working Group.
"INCOSE systems engineering handbook v. 3.2.2."
INCOSE-TP-2003-002-03.2.2. October, 2011.

[6] Charles A. Weber, John R. Current, and W. C. Benton.
"Vendor selection criteria and methods."
European journal of operational research 50.1 (1991): 2-18.

[7] Karl E. Wiegers.
"Automating requirements management."
Software Development 7.7 (1999): 1-5.

[8] Karl E. Wiegers.
"Software requirements."
Microsoft Press; 2 edition (8 Mar. 2003).

[9] Matthias Hoffmann, Nikolaus Kuhn, Matthias Weber and Margot Bittner.
"Requirements for requirements management tools."
Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International. IEEE, 2004.

[10] Azida Zainol and Sa'ad Mansoor.
"An investigation of a requirements management tool elements."
Open systems (ICOS), 2011 IEEE conference on. IEEE, 2011.

[11] Daniyal M. Alghazzawi, Shams Tabrez Siddiqui, Mohammad Ubaidullah Bokhari and Hatem S Abu. Hamatta.
"Selecting Appropriate Requirements Management Tool for Developing Secure Enterprises Software."
International Journal of Information Technology and Computer Science (IJITCS) 6.4 (2014): 49.

[12] Carey Schwaber and Peter Sterpe.
"Selecting The Right Requirements Management Tool—Or Maybe None Whatsoever."
Management (2006).

[13] Orlena Gotel and Patrick Mäder.
"How to select a requirements management tool: Initial steps."
2009 17th IEEE International Requirements Engineering Conference. IEEE, 2009.

[14] Orlena Gotel and Patrick Mäder.
"Acquiring tool support for traceability."
Software and Systems Traceability. Springer London, 2012. 43-68.

[15] Raimundas Matulevičius, Patrick Heymans and Guttorm Sindre.
"COMPARING GOAL-MODELLING TOOLS WITH THE RE-TOOL EVALUATION APPROACH."
Information Technology And Control 35.3 (2006).

[16] Mary Bone and Robert Cloutier.
"The Current State of Model Based Systems Engineering: Results from the OMG^{TM} SysML Request for Information 2009."
Proceedings of the 8th Conference on Systems Engineering Research. 2010.

[17] Louis A. Le Blanc and Willard M. Korn.
"A structured approach to the evaluation and selection of CASE tools."
Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's. ACM, 1992.

[18] Jeff A. Estefan.
"Survey of model-based systems engineering (MBSE) methodologies."
Rev A, May 25, 2007, Incose MBSE Focus Group 25.8 (2007).

[19] Jerry Banks.
"Selecting simulation software."
Proceedings of the 23rd conference on Winter simulation. IEEE Computer Society, 1991.

[20] Colin O. Benjamin and Ossama A. Hosny.
"EXSEMA-An EXpert System for SElecting Simulation Software for Manufacturing Applications."
CAD/CAM Robotics and Factories of the Future'90. Springer Berlin Heidelberg, 1991. 315-320.

[21] Mark S. Greenstein.
"CAE simulation tool selection criteria for today's ASIC designs."
Fourth Annual IEEE International ASIC Conference and Exhibit — 1991, pp. T6-1.1-T6-1.3

[22] Lesley Davis and Glyn Williams.
"Evaluating and selecting simulation software using the analytic hierarchy process."
Integrated manufacturing systems 5.1 (1994): 23-32.

[23] Jalal Nikoukaran and Ray J. Paul.
"Software selection for simulation in manufacturing: a review."
Simulation practice and theory 7.1 (1999): 1-14.

[24] Jalal Nikoukaran, Vlatka Hlupic and Ray J. Paul.
"A hierarchical framework for evaluating simulation software."
Simulation Practice and Theory 7.3 (1999): 219-231.

[25] M. A. Azadeh and S. Nazari Shirkouhi.
"Evaluating simulation software using fuzzy analytical hierarchy process."
In Proceedings of the 2009 Spring Simulation Multiconference (SpringSim '09). Society for Computer Simulation International, San Diego, CA, USA, , Article 41 , 9 pages.

[26] Shady Attia, Jan LM Hensen, Liliana Beltrán and André De Herde.
"Selection criteria for building performance simulation tools: contrasting architects' and engineers' needs."
Journal of Building Performance Simulation 5.3 (2012): 155-169.

[27] Suay Ereeş, EmelKuruoğlu and Nilgün Moralı.
"An application of Analytical Hierarchy Process for simulation software selection in education area."
Frontiers in Science 3.2 (2013): 66-70.

[28] Romain Franceschini, Paul-Antoine Bisgambiglia, Luc Touraille, Paul Bisgambiglia and David Hill.
"A survey of modelling and simulation software frameworks using Discrete Event System Specification."
OASIcs-OpenAccess Series in Informatics. Vol. 43. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[29] Stewart Robinson.
"Simulation: the practice of model development and use."
Palgrave Macmillan, 2014.

[30] Nadja Damij, Pavle Boškoski, Marko Bohanec and Biljana Mileva Boshkoska.
"Ranking of Business Process Simulation Software Tools with DEX/QQ Hierarchical Decision Model."
PloS one 11.2 (2016): e0148391.

[31] H. Rashidi.
"Discrete simulation software: a survey on taxonomies."
Journal of Simulation (2016).

[32] Anil S. Jadhav and Rajendra M. Sonar.
"Evaluating and selecting software packages: A review."
Information and software technology 51.3 (2009): 555-563.

[33] Anil S. Jadhav, and Rajendra M. Sonar.
"Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach."
Journal of Systems and Software 84.8 (2011): 1394-1407.

[34] Mohamed Sarrab and Osama M. Hussain Rehman.
"Empirical study of open source software selection for adoption, based on software quality characteristics."
Advances in Engineering Software 69 (2014): 1-11.

[35] Ian Alexander.
"Tool Vendors : List of Requirements Management Tools, Requirements Engineering Tools."
Website (last visited 2016.06.10):
`http://www.scenarioplus.org.uk/`

[36] Ludwig Consulting Services, LLC.
"Requirements management tools."
Website (last visited 2016.07.14):
`http://www.jiludwig.com/Requirements_Management_Tools.html`

[37] Volere - Atlantic Systems Guild Ltd.
"Requirements Tools."
Website (last visited 2016.06.10):
`http://www.volere.co.uk/tools.htm`

[38] Tim Weilkiens.
"Popular SysML Modeling Tools."
Website (last visited 2016.06.10):
`http://list.ly/list/23A-popular-sysml-modeling-tools`

[39] Thomas L. Saaty.
"How to make a decision: the analytic hierarchy process."
European journal of operational research 48.1 (1990): 9-26.

[40] R. Timothy Marler and Jasbir S. Arora.
"Survey of multi-objective optimization methods for engineering."
Structural and multidisciplinary optimization 26.6 (2004): 369-395.

[41] U.S. Food and Drug Administration (FDA).
"Protecting and Promoting Your Health."
Website (last visited 2017.01.24):
`http://www.fda.gov/`

[42] Christof Ebert and Capers Jones.
"Embedded software: Facts, figures, and future."
Computer 42.4 (2009): 0042-52.

[43] Diagramming.org.
"The Master Directory of Diagramming Tools."
Website (last visited 2016.07.18):
`http://www.diagramming.org/`

[44] Softdevtools.com.
"Tools directory: approach: UML Unified Modeling Language."
Website (last visited 2016.07.18):
`http://www.softdevtools.com/modules/weblinks/viewcat.php?cid=54`

[45] Modeling-languages.com.
"UML tools."
Website (last visited 2016.07.18):
`http://modeling-languages.com/uml-tools/`

[46] OMG-SysML.
"Systems Modeling Language"
Website (last visited 2014.05.26):
`http://www.omgsysml.org/`

[47] Lenny Delligatti.
"SysML Distilled: A Brief Guide to The Systems Modeling Language."
Pearson Education, Inc., 2014.

[48] Jon Holt and Simon Perry.
"SysML for Systems Engineering."
(Professional Application of Computing Series 7), IET, 2008.

[49] Tim Weilkiens.
"Systems Engineering with SysML/UML: Modeling, Analysis, Design."
Kaufmann, 2006.

[50] OMG-UML.
"Unified Modeling Language^{TM} (UML®)."
Website (last visited 2014.05.26):
`http://www.uml.org/`

[51] OMG-MARTE.
"The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems."
Website (last visited 2014.05.26):
`http://www.omgmarte.org/`

[52] Gerd Behrmann, Alexandre David, and Kim G. Larsen.
"A Tutorial on Uppaal 4.0", "Updated November 28, 2006."
`http://www.uppaal.com/admin/anvandarfiler/filer/uppaal-tutorial.pdf`.

[53] 20sim.
"What is 20-sim?", v4.4.
Website (last visited 2014.04.24):
`http://www.20sim.com/`

[54] Destecs.org.
"DESTECS (Design Support and Tooling for Embedded Control Software)."
Website (last visited 2014.04.24):
`http://www.destecs.org/`

[55] Mentor Graphics.
"SystemVision Multi-Discipline Development Environment."
Website (last visited 2016.07.18):
`https://www.mentor.com/products/sm/system_integration_simulation_analysis/systemvision/`

[56] National Instruments.
"LabVIEW System Design Software."
Website (last visited 2016.07.18):
`http://www.ni.com/labview/`

[57] Peter Fritzson.
"Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica."
Wiley, 2011.

[58] Peter Fritzson.
"Principles of Object-Oriented Modeling and Simulation with Modelica 2.1."
Wiley 2004.

[59] UC Berkeley EECS.
"Ptolemy-II domains", ptII8.1.
Website (last visited 2014.04.24):
`http://ptolemy.eecs.berkeley.edu/ptolemyII/ptII8.1/ptII/doc/domains.htm`

[60] Mathworks.
"Supported Models for Time- and Frequency-Domain Data", R2014a,
Website (last visited 2014.04.24):
`http://www.mathworks.se/help/ident/ug/supported-models-for-time-and-frequency-domain-data.html#bs24kjk-1`

[61] Mathworks.
"Supported Continuous- and Discrete-Time Models", R2014a,
Website (last visited 2014.04.24):
`http://www.mathworks.se/help/ident/ug/supported-continuous-and-discrete-time-models.html`

[62] Mathworks Matlab/Simevents.
"SimEvents - Model and simulate discrete-event systems."
Website (last visited 2014.04.24):
`http://se.mathworks.com/products/simevents/`

[63] Pierre de Saqui-Sannes and Jérôme Hugues.
"Combining SysML and AADL for the Design, Validation and Implementation of Critical Systems."
ERTS2 2012, Toulouse, France, 2012.

[64] Carl-Johan Sjöstedt , Jianlin Shi, Martin Törngren, David Servat, DeJiu Chen, Viktor Ahlsten, Henrik Lönn.
"Mapping Simulink to UML in the design of embedded systems: Investigating scenarios and transformations."
OMER4 Post-proceedings, 2008, pp. 1 37-160.

[65] Derek Piette.
"PTC MDM: Commercial Management Approach."
International Council on Systems Engineering (INCOSE), International Workshop 2013, Presentation,
`www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:incose_mbse_ws_ptc_final.pdf`.

[66] Mark Sampson.
"MBSE Workshop - Model Management in PLM - Teamcenter Model Management."
International Council on Systems Engineering (INCOSE), International Workshop 2013, Presentation,
`www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:siemens_tc_mbse_status_01-21-13v2.pdf`.

[67] Amit Fisher.
"IBM System and Software Solutions –Design and Model Management across the Product Development Lifecycle."
International Council on Systems Engineering (INCOSE), International Workshop 2013, Presentation,
`www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:ibm_model_management_-_incose_workshop_jan_26-27.pdf`.

[68] Manas Bajaj.
"SLIM for Model-Based Systems Engineering."
International Council on Systems Engineering (INCOSE), International Workshop 2013, Presentation,
`www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:slim_-_bajaj_-_incose-iw_2013-01.pdf`.

[69] Hans Peter de Koning.
"Multi-Domain Model-Based Engineering."
International Council on Systems Engineering (INCOSE), International Workshop 2013, Presentation,

www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:07b-2013_incose_mbse_workshop-multi-domain_model-based_engineering--dekoning.pptx.

[70] Darren W. Dahl and Page Moreau.
"The influence and value of analogical thinking during new product ideation."
Journal of Marketing Research 39.1 (2002): 47-60.

[71] Kees Dorst.
"The core of "design thinking"and its application."
Design studies 32.6 (2011): 521-532.

[72] C. L. Dym, A. M. Agogino, O. Eris, D. D. Frey and L. J. Leifer.
"Engineering design thinking, teaching, and learning."
Journal of Engineering Education, 2005, 94.1, 103-120.

[73] Allan Munck and Jan Madsen.
"A systematic and practical method for selecting systems engineering tools."
Accepted for presentation at 11th Annual IEEE International Systems Conference, 2017. To be published in the conference proceedings in IEEE IEL and IEEE Xplore.

[74] Mathworks Matlab/Simulink.
"Simulation and model-based design for dynamic and embedded systems."
Website (last visited 2016.01.24):
http://www.mathworks.se/products/simulink/

[75] Mathworks Matlab/Stateflow.
"Design environment for developing state charts and flow graphs tightly integrated with MATLAB and Simulink."
Website (last visited 2016.01.24):
http://www.mathworks.se/products/stateflow/

[76] Sanford Friedenthal, Alan More and Rick Steiner.
"A Practical Guide to SysML: The Systems Modeling Language."
Morgan Kaufmann; 2nd Revised edition (26 Nov 2011), ISBN-10: 0123852064.

Chapter 3

Test-driven modeling of Embedded Systems

To benefit maximally from model-based systems engineering (MBSE) trustworthy high quality models are required. From the software disciplines, it is known that test-driven development (TDD) can significantly increase the quality of the products. Using a test-driven approach with MBSE is likely to have a similar positive effect on the quality of the system models and the resulting products. To define a test-driven model-based systems engineering (TD-MBSE) approach, one must define this approach for numerous sub-disciplines such as modeling of requirements, use cases, scenarios, behavior, architecture, etc. In this chapter, we present a method that utilizes the formalism of networks of timed automatons with formal and statistical model checking techniques for applying test-driven modeling (TDM) of the architecture and behavior of embedded systems of medium complexity. The results obtained suggest that the method provides a sound foundation for rapid development of high quality system models leading to well-designed products.

The remaining parts of this chapter are divided in the following sections. Section 3.1 introduces the background of test-driven modeling of embedded systems. The characteristics of such systems are described in section 3.2. Section 3.3 describes related works. The case study concerning parts of an embedded system is introduced in section 3.4, and the proposed method is described in section 3.5. The application of the proposed method to the case is described in section 3.6. Section 3.7 discusses the results of using the method, observations, learning points, advantages, disadvantages, limitations, potential improvements and future work. The section is ended by an overall conclusion on the chapter. References of cited works are finally listed in section 3.8. This chapter extends and refines the paper [2].

3.1 Introduction

The complexity of embedded and other systems has grown rapidly during the last couple of decades and this trend is expected to continue in the future [7]. To handle this growing complexity, many organizations have progressed from document-based systems engineering (DBSE) to MBSE. MBSE provides numerous advantages compared to DBSE, including the possibility of building models that can be analyzed or simulated to get early indications of design flaws, performance issues and other problems. However, growing complexity of systems will lead to growing complexity of models in the future. Therefore, it is not difficult to project a future, where system models are much more complex than implementations of contemporary systems. Even now, some organizations work with vary large models that can be understood only by looking at smaller parts at a time. The risk of incomplete and inconsistent models is thereby growing. Means to handle the growing complexity of models are therefore needed.

The use of abstractions, layering, hierarchies, specialized view for individual stakeholders, simulations, and other MBSE techniques, has previously made it possible for system engineers to design, verify and validate complex systems. Such techniques, however, will most likely not suffice in the future when models become too complex for human minds to comprehend. Therefore, system engineers and developers need modeling methodologies, where analyzing and testing are automated and replace or complement manual model inspections, reviews, etc.

Software development is another discipline that has been challenged by size and complexity. Several techniques have been proposed and used to handle such complexity. It is known that the use of TDD can increase the quality of software significantly [8], [9], [47], [10], [49]. In this project, it is assumed that TDM similarly can increase the quality of system models and make it possible to handle large and more complex systems.

Transforming TDD into a TD-MBSE method is not straightforward. In TDD, the test-driven approach (TDA) is mainly used to ensure that the functions of the units under testing behave as expected. In TD-MBSE, the TDA can be used in a much more general way to ensure correctness of the models of stakeholders, life cycle phases, requirements and constraints, use cases, behavior including error handling, architecture and performance, data and objects, traceability, etc. While overall TD-MBSE may be desirable, it is assumed in this project that TDM of system architecture and behavior will be more valuable. The hypothesis is that the UPPAAL family of modeling languages and tools can facilitate this form of TDM for developing embedded or similar systems. This chapter therefore focuses on this limited form of TDM, while the general application of TDA to other modeling disciplines is left for future work.

3.2 Systems characteristics

Embedded systems define a wide variety of products that are:

- Detecting their environments through sensors.
- Converting inputs to outputs through processors.
- Acting in their environment through actuators.

The sensors used for detecting environments or users can include various forms of buttons, touch-screens, microphones, light detectors, cameras, magnetometers, smoke detectors, electronic noses, flow sensors, thermal sensors, bio-sensors, chemical sensors, accelerometers, etc. Sensors may include electronics and software parts for pre-processing of raw sensor data into meaningful information.

The processors used for converting inputs to outputs can include general central processing units (CPUs), sound and image processors including digital signal processors (DSPs), network processors, graphics processing units (GPU), various types of co-processors such as floating-point units (FPUs), memory systems, etc. Historical processors are described in details by Heath [6, chp. 2]. Most types of processors include both hardware and software parts.

The actuators used for acting in the environments or for informing or affecting users can include various forms of mechanical actuators using electrical, pneumatic (pressurized air) and hydraulic (liquids) forces; electric actuators using conducting currents, magnetic loops, and antennas; thermal actuators; visual actuators using lamps, light emitting diodes (LEDs), and computer screens; audio and vibrational actuators using loudspeakers, vibrators, etc. Actuators may include electronics and software for converting commands into useful actions.

The characteristics of embedded systems can be expressed in terms of the requirements and constraints to which they must conform. Wiegers [4] and the INCOSE

Systems Engineering Handbook [5] contain long lists of functional requirements (business, user and system requirements) and non-functional requirements (business rules, quality attributes, interface requirements and architectural constraints) that also apply to embedded systems.

Human life often depend strongly on embedded systems. Malfunction may harm the environment, damage objects, and injure or even kill users or bystanders. For such safety critical systems, functionality and performance must be guaranteed under adverse circumstances. Embedded systems must therefore be reliable and include some of the following functions:

- Fault detection.
- Fault forecasting.
- Fault tolerance.
- Fault prevention.
- Fault removal.
- Fault recovery.

Embedded systems are also subject to various types of timing constraints:

- Hard real-time constraints.
- Firm real-time constraints.
- Soft real-time constraints.

For the *hard* real-time constraints, failure to respond within the specified range causes total system failure. For the *firm* real-time constraints, occasional failure to respond within the specified range is tolerable, but causes useless results, and repeated failures may degrade the system. For the *soft* real-time constraints, repeated failure to respond within the specified range is permissible, but the usefulness of results degrade after missing deadlines. Real-time requirements can originate from several sources such as users, environmental systems, regulations and standards, system architecture, communication protocols, etc.

The hearing systems from GN Hearing contain several products that may be classified as embedded systems. Hearing instruments may not damage the ears of the users. Fault prevention dictates that the sound pressure level (SPL) must be limited and never exceed specified levels in presence of errors and failures under operational conditions. Timing within the hearing instruments are mainly constrained by the system architecture and the protocols used for communicating with accessories. The case described in section 3.4 contains *hard* real-time constraints caused by hardware architecture and the protocols for communicating with the constituent parts. Failure to meet deadlines may cause total system failure and damage to components.

3.3 Related work

There is a huge amount of literature relating to TDM of embedded systems. Model-driven development is discussed in section 3.3.1 and test-driven development is discussed in section 3.3.2. Different attempts at TDM are discussed in section 3.3.3. The difference between TDM and model-driven/-based testing is described in section 3.3.4. The foundations of the proposed method (tools, verification techniques, formal and statistical model checking, simulating) are described in 3.3.5. A summary of the related work is finally presented in section 3.3.6.

3.3.1 Model-driven development (MDD)

MBSE and MDD have been applied in various forms by numerous organizations.

Friedenthal et al. [14], Delligatti [15], Wielkins [16], and Holt and Perry [17] described modeling with SysML. Lavi and Kudish [18] described systems modeling with the ECSAM method. Numerous authors have described modeling using other methodologies, languages and tools.

Ambler [20] argues that model driven development (MDD) should be agile and as simple as possible. He thus rejects the idea of "generative" MDD (code generation, using model driven architecture (MDA)) due to the lack of proper modeling languages, tools and skilled developers. Models should be used as a means to think before coding. Later Ambler [21] discussed how to benefit from modeling, without suffering from the drawbacks, by using just-good-enough models, such that the obtained value versus the effort spent could be optimized. The proposed agile model-driven development (AMDD) is characterized by active stakeholder participation, collective ownership, model with others, and prove-it-with-code. AMDD is enabled by TDD and various forms of refactoring.

However, reliability and timing should be thoroughly verified for embedded systems. Verification by testing on actual hardware and software will at best find errors late in the development cycle, and there is a great risk of overlooking errors by applying

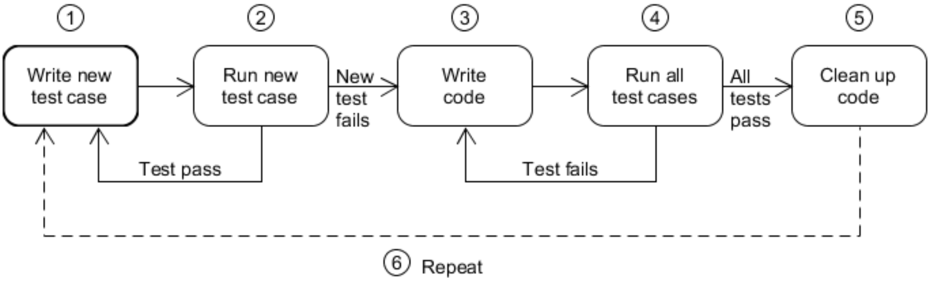


Figure 3.1: TDD for software development.

indiscriminate testing. Using models that can be executed, simulated, or verified formally or statistically is probably a better approach to guarantee the functionality and performance of embedded systems. Ambler's AMDD approach therefore cannot be used for reliable development of embedded systems. However, it can be used for preparing executable models.

3.3.2 Test-driven development (TDD)

TDD has been used in agile software development for making better customer-oriented products. TDD may also be adapted to development of better embedded systems. Several variants of the TDD method exist, e.g. [47], [48], [49]. The TDD method can be described as a process consisting of the following steps:

TDD-1 Write a new test case representing a new requirement.

TDD-2 Run the new test case to ensure that it fails.

TDD-3 Write new production code or modify existing code.

TDD-4 Run all tests until all tests are passed or until a test case fails. In case of failure, go to step (TDD-3), otherwise continue from step (TDD-5).

TDD-5 Clean up the code (refactoring).

TDD-6 Repeat the process for the next requirement, go to step (TDD-1).

This interpretation of TDD is illustrated in figure 3.1. The step of cleaning up the code (TDD-5) is optional in some variants.

Janzen and Saiedian [22] described the effect of using TDD for software development. Using TDD results in smaller and less complex units that are better tested, and the average line coverage (percentage of lines of subjected to automated testing) is shown to be significantly higher. However, claims of improved cohesion and lower coupling could not be confirmed.

Shull et al. [23] presented a review of 33 studies concerning TDD. The effect of using TDD on delivered quality was perceived as better, comparable or worse in 13, 5 and 2 studies, respectively. The effect on internal code quality was perceived as better, comparable or worse in 6.5, 4 and 3.5 studies, respectively (some studies showed mixed results, thus acquiring the grades 0.5). The effect on productivity was perceived as better, comparable or worse in 10, 6, and 9 studies, respectively. It is uncertain if the writing of test cases was included for the non-TDD cases. Maintenance of code may also not have been included in the metrics. For systems modeling, model quality is of highest importance, whereas the internal code quality has lower importance, because models often are simplified abstractions of hardware, software, users and environments, so they cannot be reused for code generation, anyway.

Grenning [24] described how to adapt and apply TDD practices to embedded software development. He distinguished between unit testing and acceptance testing (scenarios to test integrated groups of units). He defined 5 stages: (1) TDD on the development system; (2) Compile for target; (3) Run test on evaluation boards; (4) Run test in target hardware (HW); (5) Manual target testing (under realistic environmental conditions). HW is abstracted as interface classes or functions and mock-ups are used in stage 1 and 2 for allowing early development and testing of SW (before HW is available). Greening's approach mainly concerns development of the software of embedded systems, after the system requirements and design have been determined. Greening's method should therefore be preceded by TDM, where the basic system design is modeled and verified, before commencing expensive and resource demanding development of hardware and software.

Collier and Ozik [25] discussed TDD for agent-based modeling and simulating (ABMS) using specific toolkits. ABMS is typically used for analyzing so-called natural systems (biology, economics, social systems, business, networks, etc.) to gain insight into the collective behavior of agents. ABMS methods may be adapted for development of embedded systems.

Overall, it can be concluded that TDD has the potential to improve quality and productivity, and it can be adapted to other domains than those for which it was originally intended.

3.3.3 Test-driven modeling (TDM)

The idea of TDM is not new, but the translation from TDD to TDM is not straightforward. This may be one of several reasons for the limited progress in this area. However, some attempts have been reported as described in the paragraphs below.

Zhang [11] described a method for test-driven software modeling utilizing simulations of message sequence charts (MSC). Experimental data shows that this methodology can be applied to large projects resulting in high productivity and quality in terms of the number of code defects. However, his method with its focus on using MSC in a limited software domain may not scale to verification of embedded systems comprising numerous concurrent hardware activities and software processes. State machine formalisms are probably better for expressing the behavior of such systems.

Hayashi et al. [12] described a method for test-driven UML-based modeling of software systems that utilizes simulations of unit and scenario tests. They also developed a simple tool for their methodology. Model elements are created semi-automatically from failed test cases. The usability of this tool is unfortunately not at the level we are used to from more mature UML-based modeling tools. A better tool solution may contribute to a more widespread adoption of the proposed approach.

Luna et al. [26] described an approach for test-driven model-based web engineering (TDMBWE) utilizing the following steps in small increments: (1) Creating UI mockups (HTML) of the intended web applications; (2) Creating user interaction diagrams (UID) of expected behavior; (3) Writing and executing test cases (based on interaction requirements) using standard web testing tools; (4) Modeling the logic layer in WebML, incorporate the UI mockup as presentation layer, and generating the complete testable application; (5) When the tests pass, they continue with refactoring or adding new requirements. Their approach focuses on presentation or navigation layer, whereas ordinary TDD focuses on the logic layer. Later, Luna et al. [27] expanded the TDMBWE approach to include usability requirements with functional impact, without changing the fundamental approach. Their approach is thus applicable to testing of simple web application. Design and verification of complex embedded systems are beyond the scope of TDMBWE, but it may be used for developing the UI layers.

Zhang and Patel [28] described an AMDD approach for software developments utilizing UML sequence diagrams for tests and UML state machines for system models. The UML models are translated to simulation models by code generation. The case study showed a threefold productivity increase of their TDM method compared to hand coding, and increased quality (less defects). These results were obtained despite "many team members had no MDD background or prior agile experience."

Zugal et al. [29] presented a test-driven modeling suite (TDMS) to support TDM of declarative processes. Later, Zugal et al. [30] evaluated such methods empirically. They were not able to demonstrate improved model quality, but they concluded that adoption of test cases lowered the cognitive load during modeling, increased the perceived quality, and facilitated improved communication between domain experts (DE) and model builders (MB). They argued that the sequential nature of test cases provides an intuitive way of describing processes for DEs.

Mou and Ratiu [31] proposed a method for model-based test-driven systems development with seemingly limited scope. No experimental data are provided, which makes it difficult to evaluate the applicability of their method.

Dietrich, Dressler, Dulz, German, Djanatliev, Schneider, Yupatova and Deitsch developed a test-driven agile simulation (TAS) method and a complex tool chain called VeriTAS (later SimTAny) utilizing UML [40], SysML [39], MARTE [41], UPT [42], and various tools and frameworks, as described in the following paragraphs:

Dietrich et al. [32] presented an approach to validate UML models with model-level unit tests (also defined with UML). Their approach was integrated into a framework called Syntony. They applied their method to several cases and uncovered errors in existing and previously used models.

Djanatliev et al. [33] presented the VeriTAS modeling environment supporting the TAS method for software development, where system models and test models are separated and developed concurrently. The MARTE profile is used for creating systems models. Markov chain formalism is used for test models, which enable statistical test case generation. The tool chain within VeriTAS converts the system and test models to simulation code, which is executed and visualized in the tool. VeriTAS is implemented in the Eclipse RCP platform, and it integrates Papyrus UML, Enterprise Architect, .getmore, MCUEditor, Syntony, Eclipse Zest Visualization Toolkit, OMNet++ simulator, plus novel plugins developed specifically for VeriTAS.

VeriTAS was later updated by Schneider and German [34] to support distributed development teams. Model transformation, test case generation, simulation and test execution, analysis and statistical calculations are offered as services that are accessed from the VeriTAS front-ends via a ModelBus. Using distributed services is transparent to the users.

Schneider et al. [35] described how UML, SysML, MARTE and UTP are utilized in the TAS approach without suffering inconsistencies due to conflicting semantics of the applied profiles. SysML is used for requirements and high-level structural and behavioral modeling. UTP is used for describing test cases. MARTE is used for low-level modeling of non-functional properties, for defining hardware/software and for allocations. MARTE is also used for refining behavioral models.

Further development of TAS and VeriTAS was described by Schneider et al. [36].

VeriTAS is now replaced by SimTAny. It allows the user to create, verify and validate both system models and test models in small incremental steps, thus facilitating agile software development. TAS with VeriTAS/SimTAny seems to be very useful for development of embedded systems, and the included UML, SysML, MARTE and UTP profiles have many appealing features for TDM of heterogeneous embedded systems. However, the learning curves of using these profiles and associated tools are very steep. The TAS method may therefore be too cumbersome for most systems engineers. The exclusive focus on simulation rather than formal or statistical verification also limits the scope of their method. Thus, there is still a great need for TDM methods with simpler tool setups using fewer and less complicated modeling languages, which can nevertheless be utilized for accurate design and thorough verification of embedded systems.

Overall, it can be concluded that TDM has the potential to lower the cognitive load during modeling and to improve productivity and the quality of the models. However, the reviewed methods are not likely to be useful for design and verification of complex embedded systems, so new methods are needed.

3.3.4 Model-driven/-based testing (MDT, MBT)

Model-driven testing (MDT) and model-based testing (MBT) are strongly related to but different from TDM. In TDM, testing is utilized to create models of the system of interest (SOI). In MDT and MBT, modeling is utilized to create tests for acceptance or production testing of products or systems. TDM is a test-first approach, whereas MDT and MBT are model-first approaches. The experience of applying MDT and MBT has been investigated by several authors such as Amyot et al. [37] and Hartman et al. [38].

3.3.5 Tools

Modeling tools are needed to facilitate the proposed method. In the case study, three types of tools were used:

1. Tools for creating simple overview diagrams.
2. Tools for creating and analyzing models of networks of timed automatons.
3. Tools for analyzing verification results.

For overview diagrams, SysML can be utilized. Weilkiens' [46] website provides a list of popular SysML modeling tools. In this project, several SysML tools were tested including Astah, Cameo Systems Modeler (NoMagic), Enterprise Architect (Sparx), Modelio, Papyrus, Rhapsody (IBM) and Topcased. Other tools utilizing other modeling languages can also be used. In the case study, Cameo Systems Modeler was used.

For modeling and analyzing networks of timed automatons or similar representations of real-time embedded systems, several tools are available. Wikipedia [45] listed a large collection of model checking tools. The DEVINE, DREAM, RED, and UPPAAL tools utilize the timed automata formalism to describe the system under analysis, whereas other tools use different formalisms such as timed petri nets. For property checking, RED and UPPAAL use a subset of timed computational tree logic (TCTL), whereas other tools use different property languages such as linear temporal logic (LTL). Many but not all tool are targeted at real-time systems. Some tools use exhaustive model checking, whereas other use probabilistic or stochastic techniques. Several variants of UPPAAL exist. The plain version [43][13] uses exhaustive formal verification and it is targeted at real-time systems. The UPPAAL-SMC version [44] [19] uses statistical model checking and it is targeted towards a broad range of applications, including analyzing real-time systems. Other UPPAAL variants have more specialized applications. In the case study, the UPPAAL and UPPAAL-SMC tools were used. Model checking with UPPAL is further described in section 2.1 in [1]. The formalism of networks of timed automatons and the UPPAAL query language for formal model checking are described in section 2.1.1 in [1]. The formalism of networks of priced/stochastic timed automatons and the UPPAAL-SMC query languages are described in section 2.1.2 in [1].

For processing and analyzing verification results, several tools can be used such as graph programs, spreadsheets, etc. In the case study, the graph tool that is built into UPPAAL was used for creating simple graphs, and spreadsheets were used for data-fitting, extrapolating and analyzing data more thoroughly.

3.3.6 Related work summary

Modeling has been used in systems engineering as well as in software engineering, for analyzing or creating systems or products.

From the software disciplines, it is known that the use of TDD can increase the quality of software. TDD can and has been adapted to other domains such as embedded software development.

Several TDM approaches have been reported. TDM has the potential to lower cognitive load during modeling and to improve productivity and model quality. However, none of the existing TDM methods have gained widespread acceptance so far, which may be caused by several factors: All reported methods require specialized tools that may not be readily available for potential users. Some tools have bad usability. Other methods use complex tool chains that are difficult to setup and use. The modeling formalism has very steep learning curves in some cases. The methods may simple by too cumbersome to use for most systems engineers.

Existing methods are restricted to unit and scenario testing, which may fail to identify errors that emerge from combining the parts into an overall system. Without techniques to exercise the entire state-space, it is not possible to guarantee functionality and performance under all operating conditions.

Our approach to TDM differs from earlier attempts:

- It uses only existing tools that do not require integration.
- It is relatively easy to use due to the use of uncomplicated modeling methods, languages and tools.
- It includes simulations as well as formal and statistical model checking for verifying intended behavior and for capturing unintended emergent behavior.

The hypothesis is that the formalism of networks of (priced/stochastic) timed automatons with formal and statistical model checking can facilitate a TDM approach for modeling of the behavior and architectural properties of embedded systems. It is also hypothesized that UPPAAL and UPPAAL-SMC can implement such a TDM method.

3.4 Case study

3.4.1 Context

A new hardware architecture has been proposed for future hearing instruments at GN Hearing. One of the characteristics of the new architecture is that a common serial peripheral interface (SPI) bus is used for the main system, the memory system and the radio system as shown in figure 3.2. Thus, the SPI bus has become a scarce resource. The main system comprises an integrated chip with a central processor unit (CPU) and a digital signal processor (DSP). The memory system consists of a single flash memory chip. The radio system consists of an integrated chip with a wireless

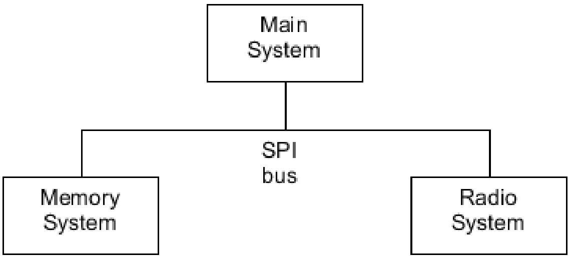


Figure 3.2: Potential new architecture to be analyzed.

radio and a local control processor. The new architecture contains numerous other elements that are not shown in the figure because they are irrelevant for the case study.

3.4.2 Problem

The problems to be investigated in the case study concern the feasibility of the new architecture, of the modeling tool, and of the proposed methodology, as outlined below:

1. Is the SPI architecture feasible?
   - Can bus conflicts be avoided and how?
   - Can time-bounded use cases be executed without failure?
   - What is the penalty on flash operations due to activities of the radio?
   - What is the penalty on radio operations due to activities of the flash?
   - What are the sensitive parameters? How are the margins?
2. Is FMC and SMC with UPPAAL feasible for the case study?
   - Is the expressiveness of the UPPAAL modeling language sufficient?
   - Will state explosion[1] prevent the use of formal model checking?
   - Is statistical model checking feasible and needed for this case?
   - Is probability estimation relevant and useful for this case?

[1] State explosion occur when the size of a state space of a system grows exponentially due to the number of processes and variables [3].

- Is value estimation relevant and useful for this case?
- Will simulation traces provide additional knowledge about the system?
- What level of expertise is needed for such UPPAAL modeling?
- Does UPPAAL provide features for easy modeling, verification and simulation?

3. Is the proposed TDM method feasible for the case study?

    - Can requirements be modeled as UPPAAL queries before creating the actual model elements?
    - Can the queries and model elements be built in small increments with or without the use of refactoring?
    - Is model debugging supported by the proposed TDM method and the modeling tool?
    - Does the proposed TDM method support design space exploration (DSE)?

The case study should be conducted such that these questions could be answered.

3.5 Proposed method

3.5.1 Basic TDM method

The proposed TDM methodology is based on TDD as described in section 3.1. The six steps of the TDD method were described in section 3.3.2. In the transformation from TDD to TDM, one must consider what to add, to modify or to remove.

The step TDD-1, *"Write new test case"*, must be modified and expanded for TDM. First, one must obtain and specify the architectural and behavioral requirements (step TDM-1), followed by a phase of refining the requirements (step TDM-2), such that they can be modeled by (semi) formal techniques (step TDM-3).

In case of behavioral requirements (i.e. required functionality), modeling in step TDM-3 is separated into two parts. First, the requirements are expressed as queries that are verified to fail before continuing. These queries mainly concern the template locations, timing and values of data. Then, the scenarios of behavioral requirements are modeled as test case templates that interact with the system templates via global communication channels and variables as shown in figure 3.3.

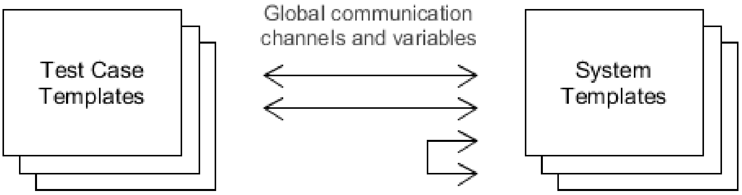


Figure 3.3: Behavioral test case modeling in UPPAAL. Test cases communicate with system templates through global channels and variables. Communication between different system templates is also facilitated by such means.

In case of architectural requirements (i.e. required quality attributes such as performance, reliability, etc.), modeling in step TDM-3 is similarly separated into two parts. First, invariants are defined as queries that are verified to fail before continuing. These queries mainly concern relationships between parameters and system properties or quality attributes. Then, the constants and variables are modeled in the templates or global declarations. See section 2.2 in [1] for further details and examples of architectural requirements modeling.

The step TDD-2, *"Run new test case"*, can be expanded for TDM. New queries are verified to fail in the model verifier (step TDM-4). Execution traces (TDM-5) may be generated for failed test cases if facilitated by the model verifier. Such traces are very effective means for driving the development of the system models as well as for the test case scenarios. Using these traces in conjunction with interactive simulations (step TDM-6) makes it relatively easy to find and correct both architectural and behavioral faults.

The step TDD-3, *"Write code"*, must be replaced by the step of modifying the system model in the TDM approach (step TDM-7). Sequences, activities, state machines, timed automatons, or similar model elements may be used. With UPPAAL, this step consists of refining the locations, transitions, and local declarations of the templates as well as the global declarations of the model.

The step TDD-4, *"Run all tests"*, is identical in the TDM approach (step TDM-8) except that we may use execution traces and interactive simulations as described above for the transformation of step TDD-2, *"Run new test case"*.

The step TDD-5, *"Clean up code"* is similar in TDM where the model may be restructured for various reasons (step TDM-9).

Table 3.1: Mapping of TDD steps and the proposed TDM steps.

Steps	TDD content	Steps	TDM content
TDD-1	Write new test	TDM-1	Obtain requirement
		TDM-2	Refine requirement
		TDM-3	Model test case
TDD-2	Run new test	TDM-4	Run new test case
		TDM-5	Execution trace
		TDM-6	Interactive simulation
TDD-3	Write code	TDM-7	Change system model
TDD-4	Run all tests	TDM-8	Run all test cases
TDD-5	Clean up code	TDM-9	Clean-up model
TDD-6	Repeat	TDM-10	Repeat

Using these transformations, a TDM method using formal modeling was developed. The relationships between TDD steps and TDM steps are listed in table 3.1. Figure 3.4 shows the overall process for the proposed TDM method. Potential minor loops and the fact that the use of execution traces and interactive simulations can be omitted is not shown in the figure.

3.5.2 Design space exploration

An important part of systems engineering is design space exploration (DSE) where alternative designs are modeled and analyzed to get the best possible solution. This part has no equivalent in the basic TDD method so DSE has to be added to the overall process.

The proposed DSE method starts by using the TDM method described in section 3.5.1 (step DSE-1) to produce a verified base design (step DSE-2).

Alternative designs are then derived from the base design by changing behavioral elements or by varying constants representing architectural properties (step DSE-3).

For behavioral variants, the sequences, activities, state machines, or timed automatons are modified in step DSE-3. The queries from the base design are mostly reused for analyzing the impact on functionality, performance, etc. However, it may be necessary to add new or to modify existing queries.

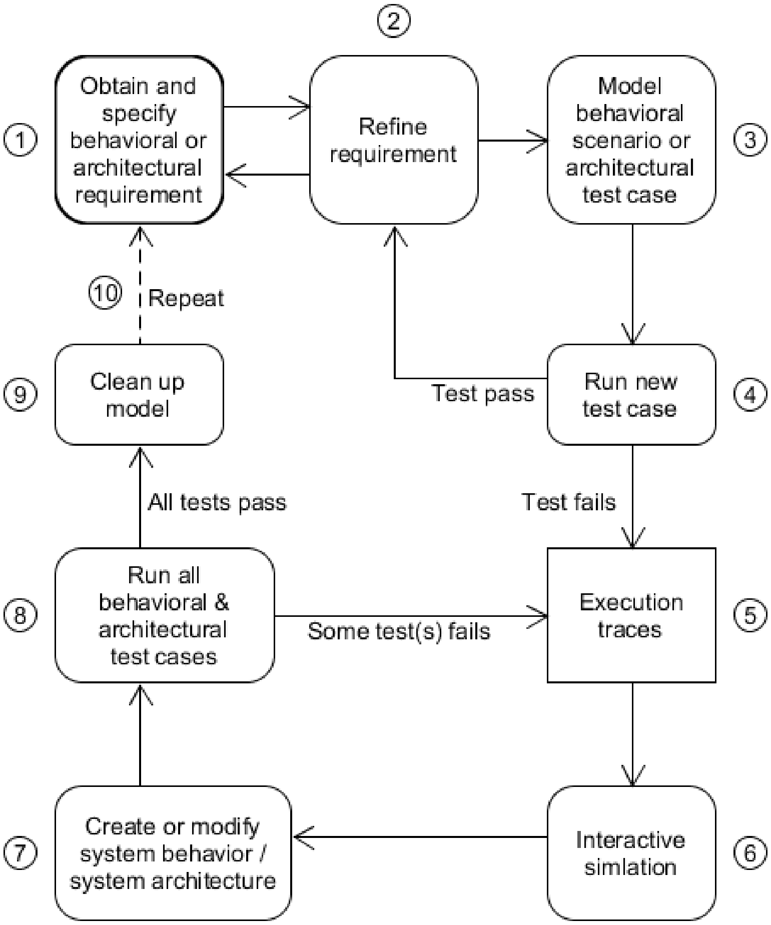

2
Obtain and specify behavioral or architectural requirement
1
Refine requirement
Model behavioral scenario or architectural test case
3
10
Repeat
Clean up model
9
Test pass
Run new test case
4
All tests pass
Test fails
Run all behavioral & architectural test cases
8
Some test(s) fails
Execution traces
5
Create or modify system behavior / system architecture
7
Interactive simlation
6

Figure 3.4: TDM for modeling of architecture and behavior. The circled numbers represent the TDM-steps.

For architectural variants, the constants that represent parameters of the model are modified in step DSE-3. Queries from the base design can be reused, but additional queries are needed for obtaining estimates of essential system properties or quality attributes (as function of the varied parameters). These estimates are obtained (step DSE-4) by using property *estimation* (step DSE-5) or *simulation* (step DSE-6), as described in section 2.1.2.2 in [1]. The parameters can be varied until satisfying estimates of the monitored system properties and quality attributes have been obtained. Thus, several iterations of step DSE-3 to DSE-6 may occur.

Satisfying estimates indicate potential solutions, but do not guarantee the feasibility of the chosen design. The preferred architectural variant must therefore be subjected to formal model checking (step DSE-7) to verify the impact on functionality (step DSE-8) and performance (step DSE-9).

Design alternatives may include both behavioral and architectural changes, because changed behaviors may cause a need for changed parameters, and changed parameters may cause a need for changed behaviors. In such cases, it may be necessary to add or redefine queries to verify design variants for guaranteed functionality and performance.

This proposed approach to test-driven design space exploration (TD-DSE) is illustrated in figure 3.5, using informal notation.

3.6 Applying method on case

The proposed method (section 3.5) was applied to the case study (section 3.4). The overall process that was used for conducting the case study is illustrated in figure 3.6, and it included the following steps:

- Collecting information:
  - Bi-weekly meetings with systems engineers to get information concerning requirements and constraints for the overall system.
  - Weekly meetings with software developers to get detailed information concerning existing use cases, protocols, timing issues, power management constraints, etc.
  - Occasional meetings with hardware developers to get detailed information concerning physical parts (processor, memory, radio) in the system.
- Processing information:

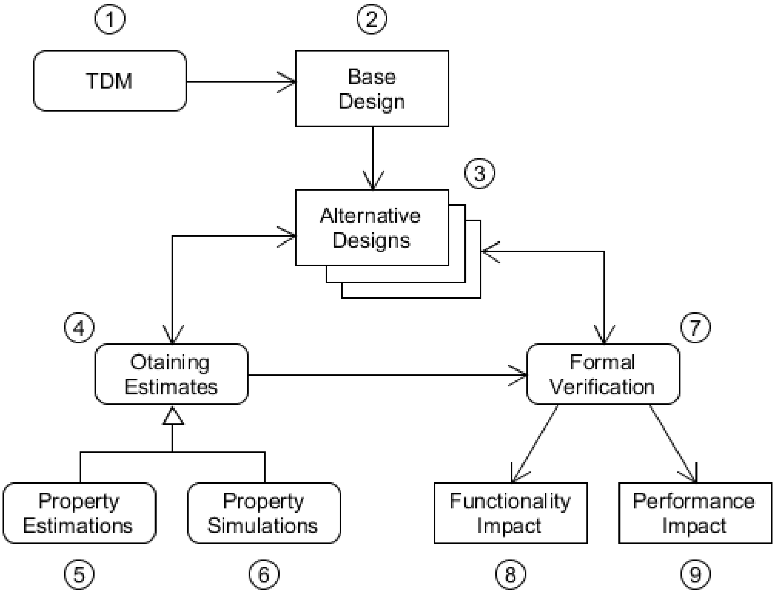


Figure 3.5: Test-driven design space exploration (TD-DSE).
The circled numbers represent the DSE-steps.

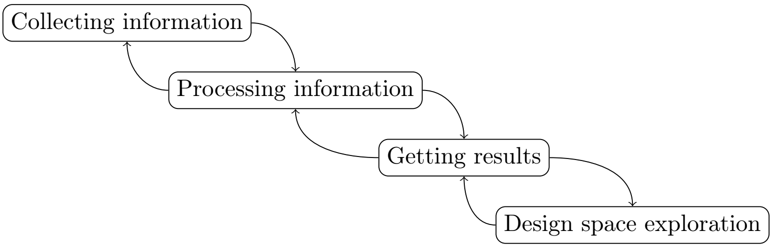


Figure 3.6: Process used for conducting case study.

- Capturing information as SysML models for structuring the system knowledge and for easier communication with stakeholders.
- Using the obtained information to create UPPAAL models according to the proposed TDM method.

- Getting results:
  - Evaluating information from SysML diagrams, tables, and generated reports.
  - Formal verification of potential design solutions using the UPPAAL models and queries.
  - Statistical simulations of UPPAAL-SMC models to get results concerning system properties and quality attributes.

- Design space exploration / alternative solutions:
  - "Efficient": Maximum utilization of the SPI bus.
  - "Fast": Minimum streaming delay and jitter (adjustable).

This process was followed by steps of evaluating the proposed method and the obtained design solutions.

3.6.1 Models

SysML was used to capture the information from the stakeholders. Section 2.3 in [1] presents a selected set of diagrams from the model. More than 30 highly interacting use cases or system functions that were potentially affected by the new architecture were identified. Several of these were constrained by strict timing requirements defined mainly by communication protocols. However, many use cases could not be executed concurrently because they require more current than the battery of the hearing instrument can deliver, so the scheduling was fairly complex and based on priorities of the functions. A worst-case combination of concurrent use cases was identified for subsequent modeling and analyzing, using the proposed TDM method. The creation of SysML models is not essential to or an integral part of the proposed method and these models will therefore not be discussed further.

The formal UPPAAL models were developed in small increments in accordance with the proposed method. These models are presented in section 2.4 in [1], where the details can be inspected. A brief overview of the two design alternatives is outlined in the following paragraphs, though.

Efficient solution

The streaming algorithm for the "efficient" solution is presented in section 2.4.5 in [1]. The `StreamingEfficient` automaton is based on the following basic algorithm:

```
1. Receive all wireless data.
2. Request access to SPI bus.
3. Wait for access.
4. Offload data to DSP.
```

This solutions has the advantages of simple implementation and of 100% utilization of the SPI bus. The disadvantage is that offloading streaming data is delayed.

Fast solution

The streaming algorithm for the "fast" solution is presented in section 2.4.6 in [1]. The `StreamingFast` automaton is based on the following basic algorithm:

```
1. Receive some wireless data.
2. Request access to SPI bus.
3. Receive remaining wireless data
   (while holding on to the SPI bus).
4. Wait for SPI access.
5. Offload data (if data is valid).
```

This solution has the advantage of minimal waiting time to access the SPI bus. The disadvantages are that the SPI bus may be blocked in vain, that it requires more complex implementation, and that the radio must support threading.

3.6.2 Model checking results

Formal model checking

The queries that were used for the formal model checking are listed in section 2.5.1 in [1] and the verification results are shown in section 2.5.2 in [1].

The "efficient" solution passed testing of all queries if the `strAllowableDelay` parameter is adjusted to the following values:

$F_{flash\ clk}$	$8 \cdot F_{min}$	$4 \cdot F_{min}$	$2 \cdot F_{min}$
`strAllowableDelay`	256 μs	512 μs	1024 μs
Queries	Pass	Pass	Pass

The formal model checking also showed that the system deadlocks when the flash clock rate is below F_{min} (company confidential). In this case, the system will not be able to comply with the timing requirements of the wireless streaming protocol.

The "fast" solution also passed testing of all queries if the `strAllowableDelay` parameter is adjusted to the following values:

$F_{flash\ clk}$	$8 \cdot F_{min}$	$4 \cdot F_{min}$	$2 \cdot F_{min}$
`strAllowableDelay`	0 μs	18 μs	530 μs
Queries	Pass	Pass	Pass

The "fast" solution also deadlocks when the flash clock rate is below F_{min}.

Two alternative design solutions have thus been obtained by using the proposed TDM method. Both solutions have been formally verified, and limits for the flash clock frequency and the maximum delay for streaming scenarios have been identified. The verification results shown that the "fast" solution causes less streaming delay than the "efficient" solution. The "fast" solution therefore *is* faster in terms of providing minimal streaming delay.

Statistical model checking

The queries that were used for the statistical model checking are listed in section 2.6.1 in [1] and the verification results are shown in section 2.6.2 in [1].

The verification results showed that the "efficient" solution uses less overhead (300 μs) in the SPI bus communication than the "fast" solution does (560 μs). The "efficient" solution therefore *is* more efficient in terms of utilizing the SPI bus.

The use of mathematical manipulations of verification results to obtain derived information (e.g. the sum of two variables) is discussed in section 2.6.2 in [1]. It was concluded that:

> *Performing mathematical operations on verification results shall be undertaken with extreme caution. Mathematical manipulations should always be performed* within *the queries and not on queried results.*

Formal verification of such expressions are not possible as explained in the appendix, so estimates can only be obtained by statistical verification or by simulation.

Simulation graphs

The queries that were used for generating the simulation graphs are listed in section 2.6.3 in [1] and the simulation results are shown in section 2.6.4 in [1].

The simulation results show that the overheads for the "efficient" and the "fast" solutions are much closer than indicated by the estimates that were obtained with statistical estimation. It was concluded that:

> *Peak values obtained from graphs with large peak-to-average ratios cannot be trusted, because peak values most likely occur between time steps rather than on the exact time of the sampling. The problem is increased by large time steps and it is decreased by small time steps. Peak values should always be verified by statistical value estimation.*

However, simulation graphs can be used for debugging the model, for inspecting system behavior, for optimizing the performance, etc. In the case study, the plots were mainly used for debugging.

3.6.3 Debugging

Models are rarely "right the first time". Normally, one must debug the models to get the models to work and to obtain the intended system functionality and performance.

UPPAAL provides the following features for debugging:

- Formal verifications:
  - Pass: Quality ensured.
  - Fail: Produces detailed traces that can be inspected.
- Interactive simulations:
  - Inspecting locations, transitions and variables during simulation.
- Statistical probability and value estimations:
  - Obtaining values to be verified by formal verification.
- Simulation plots:
  - Graphs of values for variables versus time.

From the case study, it was experienced that:

- The interactive simulations are particularly useful in the early phases to get the functionality right and to debug severe modeling errors.
- The simulation plots are particularly useful for inspecting and fine-tuning sequences of events, timing, and variable assignments.
- Statistical probability estimation is particularly useful for identifying frequently used execution paths, which can be utilized for subsequent performance optimizations.
- Statistical value estimation is particularly useful for estimating upper bounds on execution times for specific locations or paths. Such bounds should be verified by formal model checking.
- Formal verification is very useful for identifying unintended system behavior and for generating traces that can be further investigated with the interactive simulator.

The combined use of these features enabled easy model debugging in the case study.

3.7 Discussion

3.7.1 Results

The problems to be investigated in the case study were outlined in section 3.4.2.

Table 3.2 summarizes the results regarding the feasibility of the SPI architecture. It can thus be concluded that there are two feasible solutions, where the "efficient" solution gives maximum utilization of the SPI bus and the "fast" solution gives minimal delay or jitter on the streaming operations. The preferred solution depends on the acceptable levels for these effects. It is outside the scope of the case study to make that choice.

Table 3.3 summarizes the results regarding the feasibility of the UPPAAL tool. It can thus be concluded that UPPAAL is very good for conceptual modeling where formal and statistical model checking plus interactive simulations make model debugging and verification straightforward. However, it has limitations for low level modeling which may prevent it from being feasible for all potential modeling cases.

Table 3.4 summarizes the results regarding the feasibility of the proposed TDM methodology. It can thus be concluded that the system model can easily be created in small increments driven by verification of requirements expressed in the UPPAAL query language. Model debugging is greatly supported by the method and the UPPAAL tool. The design space is easily explored, and variants can be optimized by using the various model checking features.

3.7.2 Observations

The products from GN Hearing have a long legacy. Both hardware and software have evolved over time. New features have been added frequently. Existing implementations have been updated repeatedly for optimized performance of individual products. Solid engineering methods have been used to ensure sufficient quality. However, occasionally unintended behavior has been detected in early phases when products were integrated in the overall hearing systems. Dependencies from legacy code cause many such problems. This kind of problems is notoriously difficult to solve. For such reasons, investigations of systems modeling and test-driven approaches were commenced at the company.

A large range of tools were tested and tried for enabling TDM. During the tool evaluations, UPPAAL was identified as a strong candidate because it offers a unique

Table 3.2: Is the SPI architecture feasible?

Can bus conflicts be avoided and (if yes) how?	Yes, radio requests shall have highest priority. DSP algorithms compete for access to flash and they must therefore be queued.
Can time-bounded use cases be executed without failure?	Yes, if the flash clock rate is $\geq F_{min}$.
What is the penalty on flash operations due to activities of the radio?	The overhead is less than 7,3% for the "efficient" solution with $8 \cdot F_{min}$ clock rate and 4 kbyte block reads. Larger blocks yield less overhead. The "fast" solution has a maximum overhead of 13,7%.
What is the penalty on radio operations due to activities of the flash?	The streaming delay is 256 μs, 512 μs and 1024 μs for the "efficient" solution with flash clock rate $8 \cdot F_{min}$, $4 \cdot F_{min}$ and $2 \cdot F_{min}$, respectively. The streaming delay is 0 μs, 18 μs and 530 μs for the "fast" solution with flash clock rate $8 \cdot F_{min}$, $4 \cdot F_{min}$ and $2 \cdot F_{min}$, respectively.
What are the sensitive parameters? How are the margins?	The flash clock rate and the time reserved to wake up the radio are the most important parameters. The margins depend on the actual choice of several parameters. Further specifications are required to determine the margins.

Table 3.3: Is the UPPAAL tool feasible?

Is the expressiveness of the UPPAAL modeling language sufficient?	Yes, but UPPAAL does not only allow modeling of real type numbers with formal model checking. However, it is possible with statistical model checking or simulating.
Will state explosion prevent the use of formal model checking?	No, state explosion was not encountered during this case study. However, much abstraction and simplification were needed to avoid state explosion. This prevented modeling of low-level hardware behavior.
Is statistical model checking feasible and needed for this case?	Yes. Certain expressions cannot be analyzed formally or simulated with sufficient resolution. Only statistical model checking can provide reliable performance estimates.
Is probability estimation relevant and useful for this case?	Yes, it was used to estimate the probability of certain execution paths for subsequent optimization.
Is value estimation relevant and useful for this case?	Yes, it is good for estimating delays, variable values, etc. However, mathematical manipulation of results can lead to wrong conclusions. Formulas should be included in the queries and not used in post-processing of verification results.
Will simulation traces provide additional insight about the system?	Yes, and they were used to debug the model and to fine-tune the timing of the algorithms.
What level of expertise is needed for such UPPAAL modeling?	Theoretical knowledge of timed automata with cost functions are required and the UPPAAL query language must be mastered. However, the tool is not overly complex and does not require much training.
Does UPPAAL provide features for easy modeling, verification and simulation?	Yes, UPPAAL has a number of unique features that ease modeling by providing detailed traces of all execution steps, statistical estimates, and simulations graphs. Guaranteed system behavior is enabled by formal verifications.

Table 3.4: Is the proposed TDM method feasible for the case study?

Can requirements be modeled as UPPAAL queries before creating the actual model elements?	Yes, this was done throughout the case study. The model verifier generates exceptions whenever model elements do not exist, as expected.
Can the queries and model elements be built in small increments with or without the use of refactoring?	Yes, this was done throughout the case study. However, queries often have to be updated to pass verification as the model evolves.
Is model debugging supported by the proposed TDM method and the modeling tool?	Yes, the proposed method utilizes execution traces from failed tests as means for debugging. The UPPAAL tools additionally facilitate debugging by providing interactive simulations, statistical probability and value estimations, and simulation plots.
Does the proposed TDM method support design space exploration (DSE)?	Yes, several variants were created and analyzed in the case study. Two behavioral variants, "efficient" and "fast", was created. Several architectural variants concerning flash clock speed, memory block size, etc., were analyzed for both behavioral variants. All variants passed verification after adjusting the parameters for each variant.

mix of features. Using the interactive simulator with the execution traces generate by UPPAAL for failed queries made it easy to debug and update the system model. Other UPPAAL features also supported the test-driven approach.

The proposed method proved to be very effective for simple conceptual models with limited lengths of execution traces. For mixed-level model (low-level hardware models combined with long-lasting high-level scenarios), one typically has very large ratios between the longest and shortest duration of interest. In such cases, very long execution traces would be generated. However, creation of such long traces causes memory exhaustion because the entire variable space is saved for each execution step. This form of memory exhaustion was observed often for bottom-up modeling (starting at the component level). Top-down modeling (starting at the concept level) solved this problem, as described in section 3.7.3.

Section 2.7 in [1] further outlines the observations concerning the features of UPPAAL.

3.7.3 Learning points

The following important learning points were obtained from conducting the case study:

- Usable information is hard to get from existing documentation:
  - Much time was spend on identifying use cases and interactions.
  - Accurate scenario information was hard and occasional impossible to get.
  - Legacy implementations have huge impact on possible solutions.
  - Implementation details affect model elements and results significantly.
- Top-down vs bottom-up modeling approaches:
  - The first attempt was to use the component data sheets as a starting point for creating system models. Such a bottom-up approach seemed simple in the beginning. However, the required effort quickly outgrew the capabilities of the modeler and the tool.
  - The second attempt was to use a conceptual or top-down approach. This approach quickly lead to good models and interesting results.

The time spent on the different activities during the case study was not measured but loosely estimated after completion. Figure 3.7 illustrates these estimates. Most time

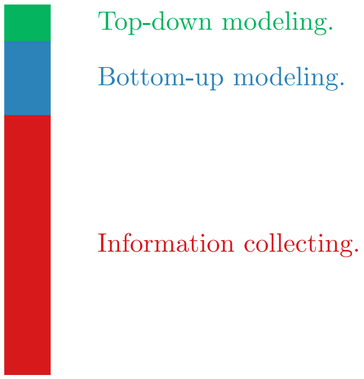


Figure 3.7: Time spent on different activities during the case study.

was spent on collecting the relevant information (red). Much time was spent on failed attempts to model the system with a bottom-up approach (blue). The conceptual modeling with a top-down approach required the least time but provided most of the results (green). It can therefore be concluded that the proposed TDM method provides quick results, if the required information is available and if the modeler is sufficiently experienced.

3.7.4 Advantages

During the case study, the following benefits of using the proposed TDM method were observed:

- System knowledge gained during modeling activities.
- Easy experimentation and design space exploration.
- Conceptual models require little information but provide good value.
- Avoiding detailed design activities on doomed concepts.

The following benefits of using UPPAAL as the principal modeling tool were observed prior to and during the case study:

- Formal verification can clarify issues that are impossible to test.
- Statistical simulations can show characteristics that are difficult to test or analyze with other methods.

- Easy model debugging due to complete traces.
- Existing SW can be included in the model (to some degree).

3.7.5 Disadvantages

The following disadvantages of the proposed method and UPPAAL were observed:

- Much abstraction and simplification are required.
- Limited expressiveness in the UPPAAL modeling and query languages.
- Parametric sweeps must be done manually.
- Careless use of statistical value estimation may lead to wrong conclusions as discussed in section 3.6.2 and section 2.6.2 in [1].
- Careless use of simulation plots may lead to wrong conclusions as discussed in section 3.6.2 and section 2.6.4 in [1].
- Low-level modeling of electronic components and high-level modeling of long-lasting scenarios cannot be combined.
- Memory exhaustion due to long execution traces.
- State explosion due to large and complex models.

The risk of memory exhaustion and state explosion constantly forces the modeler to limit the size and complexity of the models. However, the value of formal and statistical verification is weakened by the need for abstraction and simplification. Limited expressiveness also prevents models of high fidelity. These disadvantages pose some limitations on the usefulness of the proposed method.

The disadvantages are mainly caused by the limitations and problems of UPPAAL. Improving UPPAAL or identifying alternative tools may remedy these issues.

3.7.6 Limitations

Formal model checking of large models inevitably causes state explosion and memory exhaustion as discussed in section 3.7.5. The proposed method is therefore limited to modeling and analyzing relatively small and simple systems[2]. This scalability issue can be overcome to some degree by using abstractions and simplifications during modeling, but not without loss of fidelity.

The scalability problem may be resolved to some degree by modifying the memory management strategy of the tool. However, models of future systems can be expected to become very large due to increasing complexity. The need for more memory is likely to grow faster than the capabilities of future versions of the tool. The problem of state explosion and memory exhaustion may therefore persist despite improved tool capabilities.

The proposed method is limited to analyzing discrete events (DE). The DE formalism is suitable and sufficient to describe and analyze embedded software controllers. However, many types of embedded systems contain elements that cannot be described and analyzed by using the DE formalism. For example, differential equations are needed to describe electronic circuits that include energy storage elements such as coils and capacitors. Such parts can only be analyzed by using continuous time (CT) simulators. Combining DE, CT and other modes of computations into integrated heterogeneous models that can be analyzed or simulated is beyond the scope of the proposed approach.

3.7.7 Improvements

The disadvantages and limitations discussed in section 3.7.5 and 3.7.6, respectively, are candidates for potential improvements. Other potential improvements of the proposed methods are to extend the proposed TDM method for:

- Modeling of different types of systems.
- Other forms of systems modeling.

The proposed TDM method is limited to embedded systems and systems of similar complexity. However, embedded systems are currently evolving into complex smart systems, which are beyond the scope of the proposed method. A TDM method for such systems is proposed and described in section 4.6.

[2] The hearing systems of GN Hearing are generally large and complex so it could be concluded that the method is useless for the company. However, the method can be very useful for designing and verifying basic concepts or partial implementations. Furthermore, the method is included in the method for modeling and verifying large and complex smart systems as described in chapter 4.

The proposed TDM method only concerns modeling and analysis of system behavior and architectural properties relating to quality attributes. However, important metrics such as the number of interfaces, coupling between modules, etc., arise from architectural design choices that may also benefit from a test-driven approach. Modeling of user interfaces may similarly benefit from a test-driven approach. Even the modeling of stakeholders, use cases, requirements, etc. may benefit from a test-driven approach. Research concerning the general application of TD-MBSE is beyond the scope of this project, however.

3.7.8 Future work

The candidates for future work falls into the following categories:

- Better validation of the proposed method.
- Improving the proposed method and tools.
- Implementing the proposed method in the industry.

The proposed method has shown its value in the case study. However, better validation can be obtained by:

- Conducting additional case studies to evaluate the method for different but similar forms of embedded systems.
- Conducting comparative studies, where the results from using the proposed method are compared with results obtained from using other methods.
- Verification of the method by other researchers.

The proposed method can be improved to accommodate large and complex smart systems. This work has been done and is described in chapter 4. For the intended use of developing embedded systems, no potential improvements of the methodology has been identified. However, improving the tools to support the method should be part of the future work, which may include:

- Create front ends for improved usability:
  - Queries expressed in "human" language.
  - Automatic generation of UPPAAL models from SysML.

- New and updated features to improve UPPAAL:
    - "Preprocessor and compiler directives" for separating model element for formal and statistical verification.
    - Automatic creation and management of clocks for every location of the templates to facilitate easy temporal modeling and analyzing.
    - More efficient memory management (started).

Successful implementation of the proposed method in the industry requires more future work, which may include:

- Implementing TDM at GN Hearing:
    - Conduct more case studies to demonstrate the value of TDM.
    - Teach modeling skills to system engineers, software engineers, and domain experts.
    - Hire expert modelers permanently or as consultants.
    - Define and use libraries of templates and queries.
    - Integrate TDM into the overall development process.
- Implementing TDM in industry of embedded systems:
    - Conduct more research and publish scientific results.
    - Publish for target group in popular media of the industry.

Some of these activities have been started, while others may start in the near future. It is not yet determined when the remaining future work will be launched.

3.7.9 Conclusion

This chapter has addressed the need for test-driven modeling of embedded systems. The review of literature concerning related work (section 3.3) showed that:

(1) MDD has been used for development of various types of systems. MDD without testing has limited value because functionality and performance cannot be predicted, verified and validated during the modeling phase.

(2) TDD has the potential to improve software quality and development productivity. TDD has been adapted to other domains such as embedded software development. TDD has also inspired various TDM methods.

(3) TDM has the potential to lower the cognitive load during modeling and to improve productivity and model quality. Such effects have been demonstrated for both academic cases and larger industrial projects.

(4) All reviewed TDM methods use specialized and often complex tool chains with somewhat poor usability and steep learning curves.

(5) Formal or statistical verification have not been used in reported TDM methods. All reviewed methods rely on simulation frameworks.

(6) Verification to capture behavior that emerges from combining the parts of the overall system have not been addressed adequately. Existing methods focus mostly on testing of unit and system parts.

To overcome limitations of previous work a method for test-driven modeling of embedded systems has been proposed. The proposed method was developed from traditional TDD, by transforming traditional TDD steps and adding new steps that are unique for TDM of embedded systems. In addition to test-driven development of singular solutions, the proposed method facilitates exploration of the design space.

The method uses formal verification for verifying the system behavior. Statistical verification is used for estimating probabilities and values of system properties and quality attributes. Simulations are used for inspecting system states and properties.

The method was applied to an industrial case from GN Hearing for the purpose of verifying the method and for investigating potential solutions for an embedded system. Information about the system was collected into SysML models. Much time was spent on identifying and relating relevant use cases. This resulted in a large use case interaction matrix from which worst-case combinations were identified. The TDM method was utilized for developing and verifying two basic design alternatives, where an "efficient" solution gave maximum utilization of the internal bus and a "fast" solution gave minimum delay and jitter for an audio streaming scenario. It has been shown that the flash clock rate must be equal to or larger than $2 \cdot F_{min}$ (confidential) to avoid deadlock. Other critical parameters and their feasible ranges were also identified.

The proposed method is mainly limited by the use of formal and statistical model checking. The size and complexity of the models must be limited by abstractions and simplification to avoid state explosion and memory exhaustion of the verifier.

Future work has been identified for providing better validation of the proposed method, for improving the tool support, and for implementing the proposed method in the industry.

3.8 References

[1] Allan Munck
"Embedded systems models."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[2] Allan Munck and Jan Madsen.
"Test-driven modeling of embedded systems."
Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC), 2015. IEEE, 2015.

[3] Martin Kot.
"The state explosion problem", 2003.
`http://www.cs.vsb.cz/kot/down/Texts/StateSpace.pdf`

[4] Karl E. Wiegers.
"Software requirements."
Microsoft Press; 2 edition (8 Mar. 2003).

[5] INCOSE SE Handbook Working Group.
"INCOSE systems engineering handbook v. 3.2.2."
INCOSE-TP-2003-002-03.2.2. October, 2011.

[6] Steve Heath.
"Embedded systems design."
Newnes, 2002.

[7] Christof Ebert and Casper Jones.
"Embedded software: Facts, figures and future."
IEEE Computer Magazine, 2009, Volume 42, Issue 4.

[8] Piet Cordemans, Sille Van Landschoot, Jeroen Boydens and Eric Steegmans.
"Test-Driven Development as a Reliable Embedded Software Engineering Practice." In: *"Embedded and Real Time System Development: A Software Engineering Perspective."*
Springer Berlin Heidelberg, 2014. p. 91-130.

[9] E. Michael Maximilien and Laurie Williams.
"Assessing test-driven development at IBM."
Software Engineering, 2003. Proceedings. 25th International Conference on. IEEE, 2003.

[10] Dave Astels.
"Test driven development: A practical guide."
Prentice Hall Professional Technical Reference, 2003.

[11] Yuefeng Zhang.
"Test-driven modeling for model-driven development."
Software, IEEE 21.5 (2004): 80-86.

[12] Susumu Hayashi et al.
"Test driven development of UML models with smart modeling system." In: *"«UML» 2004 - The Unified Modeling Language. Modeling Languages and Applications."*
Springer Berlin Heidelberg, 2004. 395-409.

[13] Gerd Behrmann, Alexandre David, and Kim G. Larsen.
"A Tutorial on Uppaal 4.0 - Updated November 28, 2006."
`http://www.uppaal.com/admin/anvandarfiler/filer/uppaal-tutorial.pdf`

[14] Sanford Friedenthal, Alan Moore and Rick Steiner.
"A Practical Guide to SysML, Second Edition: The Systems Modeling Language."
Publisher: Morgan Kaufmann; 2 edition (October 17, 2011). ISBN-10: 0123852064. ISBN-13: 978-0123852069.

[15] Lenny Delligatti.
"SysML Distilled: A Brief Guide to The Systems Modeling Language."
Pearson Education, Inc., 2014.

[16] Tim Weilkiens.
"Systems Engineering with SysML/UML: Modeling, Analysis, Design."
Kaufmann, 2006.

[17] Jon Holt and Simon Perry.
"SysML for Systems Engineering."
(Professional Application of Computing Series 7), IET, 2008.

[18] Jonah Z. Lavi and Joseph Kudish.
"Systems Modeling and Requirements Specification Using ECSAM: An Analysis Method for Embedded and Computer-Based Systems."
Dorset House Publishing, New York, 2005. ISBN-10: 0-932633-45-5 / ISBN-13: 978-0932633453.

[19] Peter Bulychev, Alexandre Davidm, Kim Guldstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, Zheng Wang.
"UPPAAL-SMC: Statistical Model Checking for Priced Timed Automata."
arXiv preprint arXiv:1207.1272, 2012.

[20] Scott W. Ambler.
"Agile model driven development is good enough."
IEEE Software 20.5 (2003): 71-73.

[21] Scott W. Ambler.
"Agile Model Driven Development (AMDD)."
XOOTIC Symposium 2006, (p.13). 2006.

[22] David S. Janzen and Hossein Saiedian.
"Does test-driven development really improve software design quality?"
Software, IEEE 25.2 (2008): 77-84.

[23] Forrrest Shull, G. Melnik, B. Turhan, L. Layman, M. Diep and H. Erdogmus.
"What do we know about test-driven development?."
IEEE software 27.6 (2010): 16-19.

[24] James Grenning.
"Applying test driven development to embedded software."
Ieee Instrumentation & Measurement Magazine 10.6 (2007): 20-25.

[25] Nicholson Collier and Jonathan Ozik.
"Test-driven agent-based simulation development."
Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World. IEEE Press, 2013.

[26] Esteban Robles Luna, Julián Grigera, and Gustavo Rossi.
"Bridging test and model-driven approaches in web engineering."
International Conference on Web Engineering. Springer Berlin Heidelberg, 2009.

[27] E. R. Luna, J. I. Panach, J. Grigera, G. Rossi and O. Pastor
"Incorporating Usability Requirements In a Test/Model-Driven Web Engineering Approach."
J. Web Eng. 9.2 (2010): 132-156.

[28] Yuefeng Zhang and Shailesh Patel. *"Agile model-driven development in practice."*
IEEE software 28.2 (2011): 84.

[29] Stefan Zugal, Jakob Pinggera and Barbara Weber.
"Creating declarative process models using test driven modeling suite."
Forum at the Conference on Advanced Information Systems Engineering (CAiSE). Springer Berlin Heidelberg, 2011.

[30] S. Zugal, C. Haisjackl, J. Pinggera and B. Weber.
"Empirical evaluation of test driven modeling."
International Journal of Information System Modeling and Design (IJISMD) 4.2 (2013): 23-43.

[31] Dongyue Mou and Daniel Ratiu.
"Binding requirements and component architecture by using model-based test-driven development."

Twin Peaks of Requirements and Architecture (Twin Peaks), 2012 IEEE First International Workshop on the. IEEE, 2012.

[32] I. Dietrich, F. Dressler, W. Dulz and R. German.
"Validating UML simulation models with model-level unit tests."
Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (p. 66). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[33] A. Djanatliev, W. Dulz, R. German and V. Schneider.
"Veritas - A versatile modeling environment for test-driven agile simulation."
Proceedings of the 2011 Winter Simulation Conference (WSC), (pp. 3657-3666). IEEE, 2011.

[34] Vitali Schneider and Reinhard German.
"Integration of test-driven agile simulation approach in service-oriented tool environment."
Proceedings of the 46th Annual Simulation Symposium. Society for Computer Simulation International, 2013.

[35] V. Schneider, A. Yupatova, W. Dulz and R. German.
"How to avoid model interferences for test-driven agile simulation based on standardized UML profiles (work in progress)."
Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative (p. 35). Society for Computer Simulation International, 2014.

[36] V. Schneider, A. Deitsch, W. Dulz and R. German.
"Combined Simulation and Testing Based on Standard UML Models."
Principles of Performance and Reliability Modeling and Evaluation. Springer International Publishing, 2016. 499-523.

[37] Daniel Amyot, Jean-François Roy and Michael Weiss.
"UCM-driven testing of web applications."
International SDL Forum. Springer Berlin Heidelberg, 2005.

[38] Alan Hartman, Mika Katara and Sergey Olvovsky.
"Choosing a test modeling language: A survey."
Haifa Verification Conference. Springer Berlin Heidelberg, 2006.

[39] OMG.
"Systems Modeling Language."
Website (last visited 2014.05.26):
`http://www.omgsysml.org/`.

[40] OMG.
"Unified Modeling Language^{TM} *(UML®)."*
Website (last visited 2014.05.26):
`http://www.uml.org/`.

[41] OMG.
"The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems."
Website (last visited 2014.05.26):
`http://www.omgmarte.org/`.

[42] OMG.
"Formal Versions of UTP^{TM}."
Website (last visted 2016.09.27):
`http://www.omg.org/spec/UTP/`.

[43] Uppsala University and Aalborg University.
"UPPAAL."
Website (last visted 2017.01.24):
`http://uppaal.org/`

[44] Alexandre David.
"Statistical Model-Checker - New SMC Extension of UPPAAL."
`http://people.cs.aau.dk/~adavid/smc/`

[45] Wikipedia.
"List of model checking tools."
Version: 19:07, 12 September 2016.
`https://en.wikipedia.org/wiki/List_of_model_checking_tools`

[46] Tim Weilkiens.
"Popular SysML Modeling Tools."
Website (last visited 2016.06.10):
`http://list.ly/list/23A-popular-sysml-modeling-tools`

[47] Kent Beck.
"Test-driven development: by example."
Addison-Wesley Professional, 2003.

[48] Laurie Williams, E. M. Maximilien and M. Vouk.
"Test-driven development as a defect-reduction practice."
Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on (pp. 34-45). IEEE, 2003.

[49] Thirumalesh Bhat and N. Nagappan.
"Evaluating the efficacy of test-driven development: industrial case studies."
Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, pp. 356-363. ACM, 2006.

Chapter 4

Test-driven modeling of smart systems

When the size and complexity of the system under consideration outgrow the capabilities of formal model checking tools, the methods described in chapter 3 become impractical or even impossible. Methods to overcome such limitations are investigated in this chapter. It will be shown that test-driven modeling of such systems can be done by combining formal methods for the basic interactions with informal simulations of scenarios with thousands of active users. Finally, a method that allow predicting the performance of extremely large and complex systems will be presented.

The remaining parts of this chapter are divided in the following sections. Section 4.1 introduces the background of test-driven modeling of smart systems. Section 4.2 describes the characteristics of such systems. The challenges associated with modeling and verifying smart systems are derived and presented in section 4.3. Related work is described in section 4.4. The case study for evaluating the proposed method is described in section 4.5. Section 4.6 presents the proposed method for test-driven modeling of smart systems. The implementation of the proposed method that was used in the case study is described in section 4.7. Section 4.8 describes the application of the method on the case. The obtained results, observations, learning points, advantages, disadvantages, limitations, potential improvements and future work are discussed, summarized and concluded in section 4.9. Reference of cited works are finally listed in section 4.10. This chapter extends and refines the paper [5].

4.1 Introduction

Mechanical products have been enriched with embedded computer systems since the late 1960's to improve functionality, performance and perceived value. Increasing amounts of intelligence have been incorporated into embedded products ever since. Currently, there is a tendency to move the intelligence to clouds or dedicated servers in order to provide functionality and performance that are not achievable by separate embedded products. Thus, products have evolved into complex cloud-enabled cyber-physical smart (CECPS) systems.

Engineering of such systems is much more complicated than engineering simple mechanical or embedded systems, and it may benefit from using models throughout the life cycle of the system under consideration (SUC). In the early phases, simple sketches or similar simple models may suffice, [6][7]. Later, behavioral models of the overall system can be utilized to simulate, verify and validate behavioral requirements. Structural models can be used to clarify the connection between system parts, users and external environmental systems. Both behavioral, structural and parametric models can be used during design of system parts and test cases.

The complex characteristics of smart systems make it very difficult to predict and guarantee functionality, safety, security and performance. Test-driven methods has been used to improve the quality of software and to increase productivity, [8][9]. Methods that include test-driven modeling are similarly likely to allow effective development of large and complex smart systems with guaranteed qualities. However, the test-driven methods that are applied to development of simpler systems do not scale to smart systems because the modeling technologies cannot handle the complexity and size of the systems.

A method for test-driven modeling that scales to very large and complex smart systems is therefore proposed and described in this chapter. The method uses a combination of formal verification of basic interactions, simulations of complex scenarios, and mathematical forecasting to predict system behavior and performance for extremely large and complex scenarios. The method was utilized to analyze, design and develop scenarios for remote fine tuning of hearing aids for a cloud-enabled hearing system from GN Hearing. The proposed method may be adapted and further improved for future development of very large and complex smart systems in various domains.

4.2 Characteristics

There have been many proposals for a definition of smart systems. Section 2.13 in [1] describes a few of these proposals. However, most definitions of smartness do not sufficiently describe the types of systems under consideration in this chapter. A feasible solution to this problem is to provide an alternative definition. The advantage of using such a definition is that it allows a more precise description of the characteristics of the types of systems under consideration in this chapter. The disadvantage is that it may differ from definitions used in other work. However, we consider the advantages outweigh the disadvantages and therefore provide the following definition:

Definition 4.1 "A *smart system* is a system of (independent) systems (SoS) containing cyber physical (sub) systems (CPS), smart (wireless) connected products and devices (SCP), clouds or web services (WS), big-data systems (BDS), and artificial intelligence systems (AIS), that service large crowds of users."

Figure 4.1 illustrates the content of a smart system according to this definition.

The following characteristics can be associated to smart systems:

- Smart features are provided by fusing sensor inputs, user interfaces, actuator outputs, web services, cloud data, big-data algorithms, artificial intelligence, etc.
- Thousands to millions or even billions of concurrent asynchronous entities (users, devices, applications, clouds, web services, etc.) interact in an unpredictable manner.
- Big volumes of data with complex structures are generated at high but irregular speed by a variety of sources in varying incompatible formats. Data may be incomplete or corrupted, and it is shared via unreliable channels. Data may be consumed or processed in a manner that have not been intended by the producer of the data.
- Sub-systems may not behave as expected, and messages between sub-systems may be delayed, corrupted or lost.
- The independence of sub-systems may cause unpredictable changes to the overall systems architecture.

Hearing systems from GN Hearing have many but not all of these characteristics. Currently, GN Hearing is in control of all subsystems except the services used for

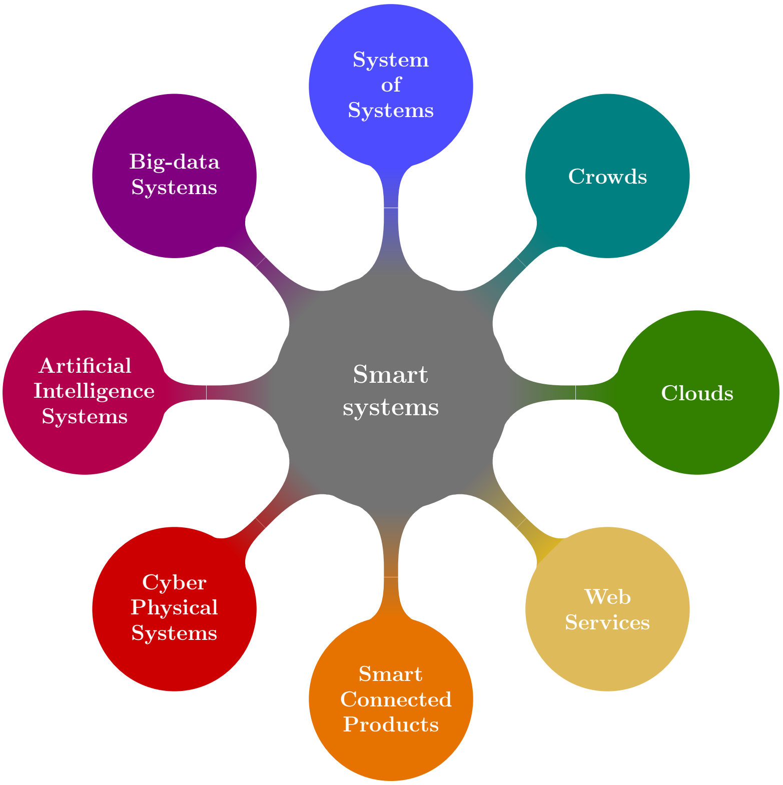


Figure 4.1: Smart system characteristics according to definition 4.1.

pushing messages to the end-users' mobile devices. The characteristics and problems associated with systems of (independent) systems are therefore less noticeable. However, GN Hearing's systems do have large crowds of users generating big volumes of data. The risk of losing data in transmissions between parts is high for such hearing systems, especially for communication between hearing instruments and mobile devices due to limited transmission power. Thus, many characteristics of general smart systems also apply to systems from GN Hearing.

4.3 Challenges

Many challenges beyond the scope of this thesis can be associated with the engineering of smart systems. However, from the characteristics described in section 4.2, the following minimal list of challenges can be derived:

- Distributing processing over several (independent) sub-systems makes it difficult to optimize the subsystems and the overall system simultaneously.
- The high level of complexity makes the design process challenging with many risks relating to maintainability, portability, extensibility, and scalability, which may compromise functionality and performance when the system evolves.
- Unintended emergent behavior may arise, when subsystems are combined. This may jeopardize the services, functions, performance and integrity of the whole system and its subsystems, which may ultimately compromise the safety of the users.

Many techniques and tools must be utilized to address such challenges. Thorough project management and rigorous systems engineering practices are the most likely means to provide good solutions. Careful requirements engineering and solid architecture descriptions provide a solid foundation upon which well-designed systems can be based. Formally analyzing and simulating system models increase the confidence in planned designs and provide means for finding and correcting errors at an early stage. Using test-driven approaches to develop system models will further increase development efficiency and product quality according to the hypotheses described in section 1.7. However, using a test-driven approach for modeling smart systems has its own set of challenges:

- Describing smart features requires modeling languages with excellent expressibility.
- Big volumes of complex data also require excellent expressibility and easy means for initializing standard or special values.
- The modeling language must also be able to express the complex behavior of subsystems, e.g. interrupt handling, database maintenance, error corrections, etc.
- Simulating or formally analyzing the system models requires very powerful engines due to the huge numbers of concurrent asynchronous entities.
- Different modes of computation may be required for different subsystems due to heterogeneity. The need for synchronizing independent simulators limits the number of available tools and modeling languages considerably.

The method that was used for test-driven modeling of embedded systems as described in chapter 3 does not scale to modeling of smart systems because:

- The modeling language has limited expressibility.

- The large number of concurrent entities will lead to state explosion.
- Large amount of data also lead to state explosion.

The method for embedded systems cannot be easily updated to accommodate smart systems. This chapter therefore proposes a novel method for test-driven modeling of smart systems.

4.4 Related work

There is a huge amount of literature relating to test-driven development of smart systems. Test-driven model-based development is mentioned in section 4.4.1. Internet of things is discussed in section 4.4.2. Section 4.4.3 discusses modeling and simulating of clouds. Simulators are discussed in section 4.4.4. Potential technologies for the proposed method are discussed in section 4.4.5 and 4.4.6.

4.4.1 Test-driven, model-based development

The work related to TDM of embedded systems is also applicable to the subject of TDM of smart systems and shall not be repeated here. The reader is referred to the descriptions in section 3.3, where MDD, TDD, TDM, MDT and MBT are described.

4.4.2 Internet of Things (IoT)

IoT systems may be considered as special cases of CECPS systems. Several researchers have investigated modeling and simulation of such systems.

Manta-Caro and Fernández-Luna [51] presented a simulator, WoTSIM, for the web of things (WoT). WoT models include spatial contexts (intelligent zones, smart spaces and sub-spaces) and temporal contexts (virtual things and virtual sensors). WoTSIM was applied to a case with 1 intelligent zone, 1 smart space, 421 smart sub-spaces, 600 virtual things and 6000 virtual sensor. Nothing indicates that WoTSIM scales to very large systems.

Brambilla et al. [24] have proposed a platform based on DEUS and OSMobility and utilizing Cooja and ns-3 for simulating large-scale internet of things (IoT) scenarios, where IoT nodes communicate, move, enter the simulation (dynamic creation) and leave the simulation (dynamic destruction). A scenario with more than 200 thousand nodes and 120 million events was simulated in less 22 hours using a server with Xeon 2 GHz CPU, 16 GB RAM, and Ubuntu Linux operating system.

Zhang et al. [25] or [26] presented an approach for automated unit testing of cloud applications. The cloud was modeled by writing stubs to simulate the real cloud. However, the cloud infrastructure was not included in the model, thus limiting the usefulness. Their approach was limited to platform-as-a-service (PaaS) computing.

Miche et al. [27] presented a concept for simulating distributed smart systems utilizing four layers (applications, overlay services, overlay networks, and underlay networks) to define their simulation models. The Common API (CAPI) (Dabek et al. [28]) defines the interface between the overlay services and networks. The modeling concept was applied to a smart aircraft-manufacturing scenario where the topology was created with the BRITE (Medina et al. [29]) tool. Actual simulations were assigned to the future work. These seems not been published, however.

The idea of using advanced network simulators such as ns-3 is also very interesting for CECPS systems because it allows precise modeling of communication between the sub-systems. However, we opted for simpler solution that can easily be applied by most systems engineers.

4.4.3 Cloud modeling & simulating

Currently there is a great interest in modeling and simulating cloud applications, architectures and deployment strategies. A total of 31 cloud simulators (CDOSim, CEPSim, CloudAnalyst, CloudSim, CloudSimSDN, DCSim, DynamicCloudSim, Ecalyptus, EMUSIM, FlexCloud, GDCSim, GloudSim, GreenCloud, GroudSim, GroundSim, iCanCloud, MDCSim, MR-CloudSim, NetworkCloudSim, ns2, OCT, OpenCirrus, OpenCloud, OpenStack, Opnet, PICS, secCloudSim, SimIC, SmartSim, SPECI, and TeachCloud) were investigated as potential tools in [30], [31], [32], [33], [34], [35], [36], and [37]. The simulators were characterized in terms of provider, underlying programming languages and platforms, simulator type (event-based or packet-level), cost modeling features (yes, no), communication modeling features (full, limited, no), energy modeling features (yes, no, rough), federation modeling features (yes, no), network modeling capabilities (limited or full), TCP/IP modeling support (none or full), GUI support (yes, no, limited), simulation time (seconds, minutes), services (IaaS, PaaS, or SaaS), implementation category (software, hardware, both), software category (software framework, simulation software, or scientific testbed), operating systems, license, availability (open source, commercial, limited, not available) and popularity.

Buyya et al. [38] and Calheiros et al. [39] described the need for cloud simulation tools and presented the CloudSim toolkit. The design, implementation, and usage of CloudSim were briefly described. CloudSim was based on GridSim, which in turn was based on SimJava.

Calheiros et al. [40] presented CloudSim 2.0, in which the SimJava was replaced by a new event management framework called CloudSim Core to overcome limitations in SimJava. The new version provides better features (starting, pausing, resuming and stopping of entities or the entire simulation) and performance (because multi-threaded SimJava was replaced with the single-threaded CloudSim Core).

Calheiros et al. [41] presented EMUSIM, which utilizes emulation of real cloud computing applications to generate simulation models that can be used for simulating different scenarios, environments, etc. Emulation is somewhat slow and only a limited number of cloud requests can be emulated. However, the request rate can be increased significantly in simulation models, because simulating is much faster than emulating. Deployment of cloud applications can therefore be tested and optimized without using real cloud infrastructures.

Wickremasinghe et al. [52] developed the CloudAnalyst tool for modeling and simulating cloud applications. The tool provides easy means for analyzing the effects of varying the deployment configurations (geographical locations, virtual machines, service brokers, etc.) on costs, response and processing time, etc. Thus, the tool is suitable for initial analyses of cloud scenarios. Detailed design is out of scope, though.

Kathiravelu and Veiga [49] and [50] presented Cloud2Sim as an extension to the CloudSim simulator. Cloud2Sim distributes simulation tasks to distributed computer nodes. Simulations that involve time-consuming processes are accelerated significantly compared to the simple sequential execution on CloudSim. However, the distribution causes overheads that renders the approach ineffective for simulating systems with processes that are less time-consuming to compute.

Other aspects of cloud modeling were addressed by Perez and Rumpe [42] (CloudADL used with CloudArc), Caglar et al. [43] (DSML used with e.g. CloudSim), Bergmayr et al. [44] (CAML UML extension & 10 reusable templates), and Fleck et al. [45] (pattern-based multi-objective optimization of cloud applications).

Cloud architecture and deployment configurations are of course important for CECPS systems. However, in our work, we are mainly interested in the overall system. For valid systems simulations, cloud models must reflect the functionality and performance of the intended cloud systems. The details of the architecture and deployment configuration are less relevant for the development phases that are currently addressed by the proposed method. The cloud simulation frameworks are of interest for future work on our method, however.

4.4.4 Large-scale simulators

Realistic scenarios of CECPS systems include huge crowds of users, devices and other sub-systems. Most simulators cannot handle systems of this scale. However, some researchers have investigated large-scale simulators.

Looga et al. [46] proposed a platform, MAMMotH, for massive-scale emulations of internet-of-things scenarios. The goal was to support up to 20 million node instances. However, development of MAMMotH seems to have stopped in 2013 [48], after its author pursued other interests [47].

D'Angelo et al. [48] proposed a multi-level simulation technique for overcoming scalability issues of existing tools to simulate IoT. Details concerning design, implementation and availability of the simulator are still to be published.

The VeriTAS/SimTAny (previously discussed in section 3.3.3) and Cloud2Sim (previously discussed in section 4.4.3) tools support distributed computing to enable large-scale simulations. However, their approach may not always be beneficial due to the overhead needed for distributing the workload. The overheads have little effect in case of heavy workloads (e.g. image processing), but overheads have severe impact on simulation time in case of many small tasks.

Good solutions for large-scale simulators need more research. In our work, we use mathematical extrapolations based on simulations of smaller scenarios to obtain estimates for large-scale scenarios. This method may not provide extremely accurate and trustworthy predictions. However, it may suffice until good large-scale simulators become widely available.

4.4.5 Technologies

The proposed process for test-driven modeling of smart systems requires a set of tools for creating descriptive models, a set of tools for creating and verifying formal models, a set of tools for creating and verifying scenario models, and a set of tools for data fitting and mathematical forecasting.

Processes, methods, languages and tools for descriptive fundamental modeling and for test-driven modeling with formal models are previously described in section 2.5, 2.7, 3.3.5, and 3.5. The discussions concerning these elements shall not be repeated here. However, it should be mentioned that SysML was selected for descriptive modeling and UPPAAL was selected for formal modeling in the case study in this chapter.

There are many potential modeling languages and tools that can be used for creating and verifying scenario-simulation models according to the proposed test-driven approach, including:

- SystemC [14][15]
- VDM/VDM++/VDM-RT/VDM-10 [16]
- SimJava2 [17]
- JavaSim (not to be confused with SimJava) [18]
- C++Sim [19]
- JSIM [20]

It is likely that all of these technologies can be used with the proposed method. Other simulators cited in section 2.2.4 may also be feasible. The SimJava2 framework was chosen for the case study, because it provides the desired features without being overly complex. Further, it is relatively easy to modify SimJava2 and tailor it for the proposed method.

There are also many potential tools that can be used for mathematical forecasting, such as spreadsheets, programming or scripting languages (e.g. Matlab, Octave, and Python), etc. Simple spreadsheets were used in the case study for convenience.

4.4.6 Simjava2

SimJava2 is well described on the official website [17]:

> "SimJava is a discrete event, process oriented simulation package. It is an application programming interface (API) that augments Java with building blocks for defining and running simulations. The original SimJava was based on HASE++, a C++ simulation library. HASE++ was in turn based on SIM++.
>
> The approach to simulating systems adopted by SimJava is similar to other process based simulation packages. Each system is considered to be a set of interacting processes or entities as they are referred to in SimJava. These entities are connected together by ports and communicate with each other by passing events. A central system class controls all the threads, advances the simulation time, and delivers the events. The progress of the

> simulation is recorded through trace messages produced by the entities and saved in a file.
>
> As of version 2.0, SimJava has been augmented with considerable statistical and reporting support. The modeler has the ability to add detailed statistical measurements to the simulation's entities and perform output analysis to test the quality of the collected data. Furthermore much effort has gone into the automation of all possible tasks allowing the modeler to focus on the pure modeling aspects of the simulation. Automated tasks range from seeding the random number generators used in the simulation, to producing detailed and interactive graphs."

It should be noted that it is possible to communicate between entities by passing events without using ports. To facilitate easy modeling in this manner, we made some minor changes and additions to SimJava2 as described in section 4.7.

More information about SimJava2 can be found on the official website [17]. SimJava is freely available and can be acquired from the download website [21]. The SimJava2 API is documented at the specification website [22]. A tutorial is available from the SimJava tutorial website [23]. Parts of the tutorial are discussed in section 4.7.

4.4.7 Related work summary

Several test-driven modeling approaches have been reported but they have not gained widespread acceptance due several factors such as poor usability, need for specialized and complex tool chains, demanding modeling formalisms, steep learning curves, etc. Previous work focuses mainly on unit or scenarios testing, whereas the proposed method concerns the entire interconnected system such that emergent behavior can be analyzed. Most prior work uses complicated and special-designed tool chains, whereas the proposed method uses simple, independent and existing modeling languages and tools that can be exchanged individually without changing the methodology.

IoT systems have many of the same characteristics as smart systems and several methods for modeling and simulating such IoT systems have been proposed. The idea of using network simulators is appealing even though the simpler and more generic SimJava2 framework was chosen for the case study in this chapter.

Smart systems are likely to include cloud elements, and much work has been reported concerning modeling and simulating cloud infrastructures and configurations. While the details of cloud architecture and deployment are less important for the overall systems modeling, the cloud simulation frameworks are of interest for future work on the proposed method.

Simulating realistic smart system scenarios requires large-scale simulators. Several methods and simulators have been proposed and reported. However, more research is needed to provide good simulators. Mathematical forecasting techniques were chosen for the case study in this chapter due to lack of feasible simulators.

Our work contributes to the field of test-driven modeling by providing a method that:

- Scales to very large and complex CECPS systems.
- Focuses on the overall interconnected systems rather than separate units.
- Captures both intended and unintended emergent behavior.
- Uses formal verifications of basic interactions.
- Uses simulations for verification of overall scenarios.
- Uses mathematical forecasting for predicting results for scenarios that are too large to be simulated.
- Uses simple, independent and exchangeable tools.
- Is easy to learn because it mainly requires basic programming skills.

For the case study, the following tools were chosen: SysML (for overview models), UPPAAL (for formal verification of basic interactions), SimJava2 (for simulating the overall system models) and spreadsheets (for data fitting and mathematical forecasting).

4.5 Case study

The proposed method was evaluated by conducting a case study where the method was applied (see section 4.8). The case study concerns an industrial case from GN Hearing. The aim of the company was to develop a cloud solution that supports remote fine tuning, remote firmware updating and usage logging features as described in section 2.11 in [1]. However, only the main scenario of the remote fine tuning feature was included in the case study because it was simple enough to demonstrate the method and complex enough to uncover the challenges. The schedule of the case study also prevented inclusion of additional scenarios.

The system under consideration in this case is further described in section 4.5.1. Section 4.5.2 describes the main scenario of the remote fine tuning feature and section 4.5.3 describes the randomness of the scenario. The goals of the case study are described in section 4.5.4.

4.5.1 System under consideration

An example of hearing systems from GN Hearing was previously shown in figure 1.3. This system provides a large collection of features and functions that have been added and accumulated over time. It has thus evolved from simpler systems in several development stages. The current developments cycle focuses particularly on adding cloud-based features. The parts of the hearing system that relate to the remote fine tuning feature constitute the system under consideration (SUC) for the case study.

Figure 4.2 shows the basic parts of the system for the remote fine tuning feature. The system may have up X million (confidential) end-users (EU). Each EU typically has one set of hearing instruments (HI) and one mobile device/application (APP), but the number of HIs and APPs can range from a one to many, independently. Each EU is associated with exactly one hearing care professional (HCP). However, each HCP may service up to hundreds of EUs. Each HCP typically has one fitting station with one fitting software (FSW) installation, but multiple stations and/or multiple FSW installations may also be possible. The typical numbers and their ranges are summarized in table 4.1.

Figure 4.2 also shows the major lines of communication in the system (many details and interactions that are irrelevant to the remote fine tuning feature are excluded from the figure). We assume that the communication between the EUs and the APPs always succeed. This assumption also applies to the communication between the HCPs and FSWs. The communication with the Cloud is conducted over the internet, which may fail, so there is no guarantee that sent messages will arrive at the destination. The communication between the HIs and the APPs is highly volatile, because there is a high risk of degraded signal quality due to path loss[1] and interference from other unrelated devices operating in the same frequency band (2.4 GHz). Therefore, this communication may fail. In addition to the potentially failing communication links, the acting system entities (Cloud, APPs, FSWs and HIs) may fail to receive, process, or send messages due to internal errors.

4.5.2 Main scenario

The main scenarios under consideration concerns a remote fine tuning features that allow HCPs to fit the HIs of EUs remotely via the internet. For this scenario, we assume as a pre-condition that all HCPs have received a request for generating a new remote fine-tuning (RFT) package for all HIs of every EU in the systems. The

[1] Increasing the distance between sender and receiver increases the path loss, which causes reduced signal to noise ratio at the receivers, leading to bit errors or loss of entire data packages.

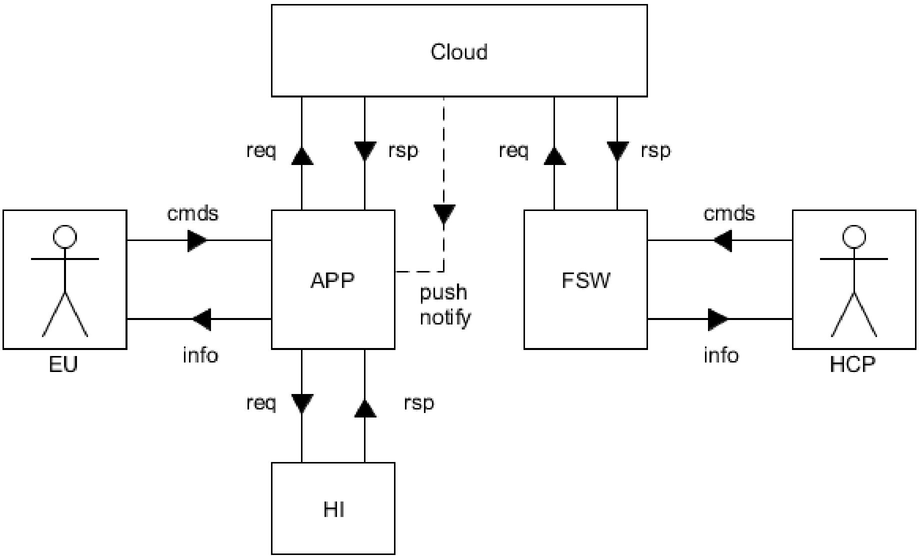


Figure 4.2: Basic parts and interactions for the remote fine tuning feature. The system may include millions of end users (EU), mobile devices/applications (APP) and hearing instruments (HI). The system also includes one cloud entity and it may include thousands of hearing care professionals (HCP) and fitting software (FSW) stations.

Table 4.1: Typical, minimum and maximum size of the system under consideration. The actual expected system size X ∈ $\mathbb{R}^+$ is confidential.

Entities	Typical	Minimum		Maximum
EUs	Up to X millions	250.000		X millions
HCPs	1 per 50-200 EUs	5.000		20.000
APPs	1 per EU	250.000		X millions
HIs	2 per EU	500.000		2·X millions
FSWs	1 per HCP	5.000		20.000
Cloud	1	1		1
Total entities (approximately)		1.000.000	to	4·X millions

main scenario can therefore be considered as a worst-case situation. The scenario to be simulated and analyzed concerns reliable transmission of remote fine-tuning packages (RFPs). It was developed specifically for simulation purposes and it does not necessarily reflect the final system developed by GN Hearing. It includes the following steps:

(a) The HCP commands the FSW to send a RFP for a specific HI and EU.

(b) The FSW sends the RFP to the Cloud and expects a confirmation response within a specified time range. The RFP will be re-send if the time limits are exceeded.

(c) The Cloud saves the RFP in its databases.

(d) The Cloud sends a confirmation response back to the FSW.

(e) The Cloud sends a notification to the APP of the EU, if the EU has enabled the Push Notification feature in the APP.

(f) The APP polls the Cloud regularly for available RFPs, if the EU has disabled the Push Notification feature in the APP.

(g) The APP informs the EU about the new available RFP.

(h) The EU chooses either to ignore or apply the RFP to the HI.

(i) The APP send the RFP to the HI, if the EU chose "apply" in the previous step.

(j) The HI may succeed or fail to apply the new RFP and responds accordingly back to the APP.

(k) The APP sends the status (applied, ignored, or failed) to the Cloud by issuing status update requests, and it expects a confirmation response within a specified time range. The status will be re-send if the time limits are exceeded.

(l) The Cloud updates the status fields in its databases and sends confirmations back to the APP.

(m) This scenario is repeated until all RFPs are stored in the Cloud and have reached the "applied", "ignored" or "failed" status.

The scenario applies to all HCPs and EUs.

4.5.3 Randomness

There is a high level of randomness in the scenario as indicated in section 4.5.1. The following unpredictable behavior can be identified for step (a) of the scenario:

- The HCPs may commence the scenario in random order and at any time.
- The HCPs may select the EUs to service in random order.
- The HCP may select the HIs of the selected EU in random order.

For step (b) of the scenario, the following forms of randomness can be identified:

- The FSWs may fail to send the message to the Cloud due to internal errors.
- The message may get lost during transmission.
- The Cloud may fail to receive the message due to internal errors.
- The FSWs may fail to re-transmit messages when needed.
- The FSWs may fail to stop re-transmissions when no longer needed.

By inspecting the main scenario, many similar forms of randomness can be identified. The model shall be able to handle all error scenarios arising from this randomness. The numbers of permutations caused by the randomness make manual inspection infeasible, though. Thorough formal analysis is neither feasible due to the size and complexity of the system. Simulating test cases to verify reliable transmission of data packages seems feasible, though. The case is therefore a good candidate for evaluating the proposed method.

4.5.4 Goals

The goals of conducting the case study were defined as:

1. Evaluate the proposed method.
2. Verify the applicability of the chosen modeling technologies.
3. Develop viable solutions to the design problem for the case.

Table 4.2: Mapping TDD steps to steps of the proposed method.

TDD steps	Proposed steps	Unique steps
	Step 1	Create descriptive models
	Step 2	Setup simulation model
Add test	Step 3	
Run test	Step 4-7	
Write code	Step 11	A, B & C
Run tests	Step 4-7	
Refactor code	Step 8 → step 11	
Repeat	Step 8 → step 3	
	Step 9	Parameter variation
	Step 10	Mathematical Forecasting

The goal of developing viable solutions was further outlined:

4. Verify absence of deadlocks.
5. Verify that stored data always have the correct values.
6. Verify that messages, events and data are routed correctly.
7. Estimate processing and turn-around times.
8. Estimate cloud loading (number of events, queue size, etc.).
9. Explore the design space for alternative solutions and compare their individual performance based on simulations and mathematical forecasting.

4.6 Proposed method

The proposed method for test-driven modeling of smart systems was developed from traditional TDD (by transforming TDD steps and by adding new unique steps) using the modeling approach discussed in section 2.1 in [3].

The method is illustrated in figure 4.3. The mapping between TDD steps and the steps of the proposed method is shown in table 4.2. The major steps of the method is further described in following paragraphs.

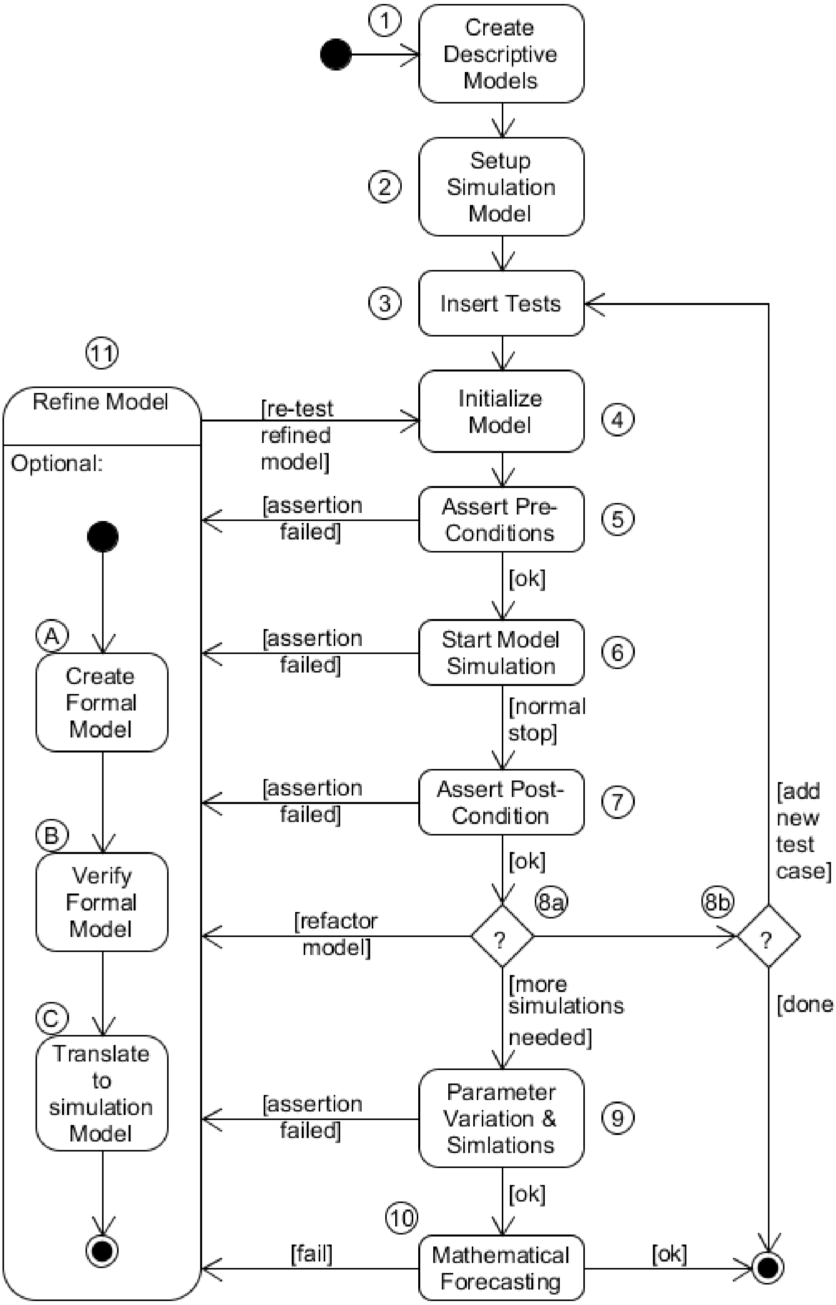


Figure 4.3: Test-driven modeling process for smart systems.

Step 1 is to create descriptive models to get an overview of the use cases, structure and interfaces of the SUC, its users and environmental systems. An agile approach without adding too many details is sufficient. These models can be created using SysML, UML, white-board drawings or similar means. Overview models are not necessarily updated when the scenario model evolves, so they may be treated as throw-away-models. After completing the overview models, go to step 2.

Step 2 is to setup the simulation model that facilitates TDM. In theory, it is possible to use any kind of modeling language and tool that supports assertions and allow execution of simulations. The SimJava2 simulation framework was used in this work due to its well-known properties and because it can be modified easily as described in section 4.4.5. The main class of the simulation is created and stop conditions are defined to ensure that the simulation can be stopped in a controlled manner. This can be done by specifying the maximum simulation time or by defining boolean expressions based on the states or data of the model. Classes that represent acting entities (users, devices, clouds, etc.) are created. Strongly related classes (e.g. users and their devices) are associated, see chapter 5 in [3] for further details. Classes for various data types are also created. Finally, an enumeration for identifying message events is created. After completing the setup, go to step 3.

Steps 3 is to insert new test cases. In ordinary TDD, all test steps and assertions would be implemented in special test units that are separated from the units of the software under testing. In TDM, it is difficult to separate the SUC and testing because the scenarios are integral parts of the SUC. Scenario steps are therefore added to the behaviors of entities that represent users or environments. Three kinds of assertions are added to enable checking of correct execution: (1) Pre-condition assertions are checked before starting the simulation. (2) Invariant assertions check the state of the model during simulation. (3) Post-condition assertions check the state of the model after completed simulation. Pre- and post-condition assertions are typically added to functions in the main program. Invariant assertions are inserted into the entities where the invariants are expected to hold. Special entities may be created just for monitoring and testing. The first failed assertion stops the simulation by generating an exception. After adding the new test cases, go to step 4.

Step 4 is to start the execution by instantiating the model. All classes necessary to describe the system, users and environments are instantiated and associated. Initializing the classes ensures that the pre-conditions can be asserted successfully. The model and simulation framework are initialized but not started yet. The source code of the simulation model must be structured such that pre-conditions are evaluated after initializing the model (go to step 5).

Step 5 is to assert the pre-conditions to ensure that the model is in the correct state before starting the simulation. Initially there is nothing to check but after several iterations, there might be data elements that must be checked. Normally, pre-conditions pass and the simulation can be started (go to step 6). In case of failure, the model must be refined (go to step 11).

Step 6 is to start the simulation. During simulation, an invariant assertion may fail. This should always happen initially in TDM; otherwise, the new test case is redundant. In case of failure, the model must be refined (go to step 11). Else, the simulation will stop normally by reaching the stop condition or by exceeding the maximum simulation time limit that has been defined in previous steps (go to step 7).

Step 7 is to assert the post-conditions to ensure that the model has entered the expected state after termination of the simulation. The data elements of the model are thus checked to verify that they have been routed correctly between entities (e.g. download from cloud). Various variables representing system states may likewise be evaluated. In case of failure, the model must be refined (go to step 11). In case of success, go to step 8.

Step 8 is to decide how to proceed after successful completion of the simulation. There are four cases as indicated by arrows out of the decision nodes (8a) and (8b) in figure 4.3. In the first case, [done], the process will be terminated because results are satisfying and no more simulations are needed. In the second case, [refactor model], the model will be cleaned up (go to step 11). In the third case, [add new test case], another test case or scenario will be added (go to step 3). In the fourth case, [more simulations needed], additional simulations of the same basic model but with modified parameters (constants within the model) are needed (go to step 9).

Step 9 is to execute the model with different values of certain parameters to get simulation results for different system configurations, reaction patterns, etc. Parameters that typically would be varied concerns the number of entities (users, devices, services, etc.), the size and amount of data, communication error rates, and probabilities of various events including user choices. Assertion for pre-conditions, invariants, and post-conditions may fail due to modified parameters, in which case the model must be refined (go to step 11). In case of success, continue by analyzing the results (go to step 10).

Step 10 is to analyze the simulation results and to forecast results mathematically for scenarios that cannot be executed in reasonable time. Data points for each variable and parameter of interest are plotted. Data points are fitted by approximate functions. If extrapolations are needed, the fitting functions must fit the data beyond the simulated scenarios, which most likely requires exponential functions. If the results obtained from interpolation or extrapolation are satisfying, [ok], the process terminates. If the results are unacceptable, [fail], the model must be refined (go to step 11).

Step 11 is the step of refining the model. This step can be reached in several cases: Failing assertions, adding new test cases, obtaining unacceptable results, or choosing model refactoring to achieve better readability, maintainability, simulation efficiency, etc. System functionality and the behavior of users and environmental systems must be implemented in the `body()` function of the relevant entity classes[2]. Communication between entities must be implemented by sending messages with associated data. Normally, one works directly in the scenario simulation model. However, sometimes it can be difficult to arrive at satisfying solutions, because the overall model can be difficult to comprehend in its entirety, especially due to concurrent activities. Therefore, it may be beneficial to utilize the optional steps to create (step A) and verify (step B) formal models of basic interactions to obtain solid patterns, before translating and integrating the solutions into the overall simulation model (step C). Using the TDM method for embedded systems that was proposed in section 3.5 is a convenient way to develop the patterns in a test-driven manner. The formal models are not necessarily updated when the scenario model evolves, so they may be treated as throw-away-models. All changes to the scenario simulation model should be made in small increments to avoid failing assertions and thus breaking the model, because it can be difficult and time-consuming to repair such complex models. After every small set of refinements, re-test the model (go to step 4).

The model is thus build in a test-driven manner. Initially, the model size (number of users and related entities, and size of data) can be limited to enable fast simulations. The size of the model can be increased later for simulating scenarios that are more realistic. During parametric variation one wants to come as close as possible to the realistic scenarios. Simulations should always be preferred over forecasts, so step 9 and 10 are only used when simulating realistic scenarios becomes impractical.

[2] The `body()` function contains the principal behavior of the entity class and is called automatically by the SimJava2 simulation framework when the simulation is started. Other simulation frameworks may use different names for such functions.

4.7 Implementing the method

The method that was proposed and described in in section 4.6 provides a blueprint for test-driven modeling of smart systems. The implementation of the method depends on the actual choice modeling technologies, i.e. the modeling languages and tools.

The implementation that was used in the case study is described in the following sections. Section 4.7.1 describes how descriptive models are implemented in SysML and section 4.7.2 describes how formal models of basic interactions are implemented in UPPAAL. How scenario simulation models are implemented, is described in the following sections: The modifications of and extensions to the SimJava2 framework are described in section 4.7.3. Section 4.7.4 describes how SimJava2 traditionally is used. To simplify modeling, ports can be avoided as described in section 4.7.5. Section 4.7.6 shows examples of using the modified and extended framework.

4.7.1 Descriptive models

One purpose of creating descriptive models is to gain an understanding of the system under consideration. Early experiments showed that it could be difficult to comprehend and maintain an overview of complex scenario-simulation models as they evolve. A second purpose of the descriptive models is to provide this overview such that the simulation models can be developed and maintained easily.

Descriptive models are not subjected to simulations but are cases of fundamental modeling (see section 2.5.1). To gain an understanding of the system under consideration, it may be beneficial to model stakeholders, life cycles, requirements, environments, users, system structure, data structure, system behavior, interfaces, parametric dependencies, etc. Examples of such fundamental modeling are provided in section 2.2 in [2].

SysML was used in the case study for creating descriptive models for the reasons described in the case study for selecting modeling tools (section 2.7 in [3]).

To provide an overview of the simulation model, it is beneficial to model the basic blocks of the simulation model on SysML block definition diagrams, BDD. Interactions and communications between entities can also be modeled on SysML BDD or more elaborately on SysML internal block diagrams, IBD. Figure 4.4 shows an example of a descriptive overview model where the entities and their interactions are modeled on a single SysML BDD. In this case, lightweight annotations (text comments) were used to indicate the tags or types of messages that would be exchanged between entities.

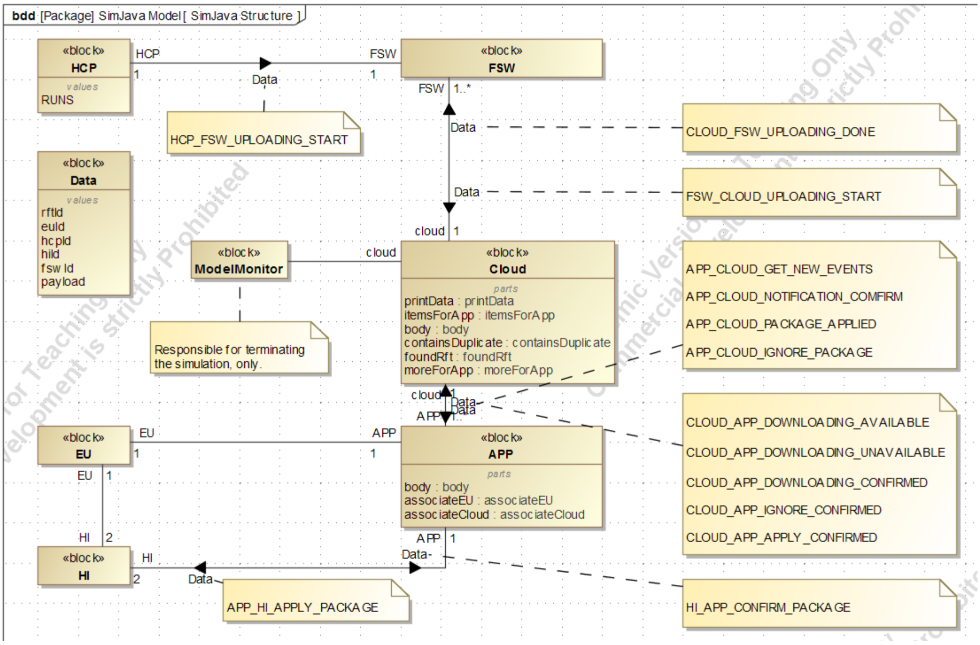


Figure 4.4: Descriptive model of simulation model showing a SysML block definition diagram depicting the interactions between the entities of the simulation model. The comments on the right are lightweight means of indicating the types of messages exchanged between the entities.

Behavioral SysML diagrams of activities, sequences or state machines were not used in the case study but such models can added to easy the subsequent development of the simulation models.

4.7.2 Formal models

The UPPAAL modeling language and tool were used for formal modeling in the case study for the reasons described in the case of selecting modeling tools (section 2.7). The formal models were developed using the test-driven method described in chapter 3.

Formal models were not made for all parts of the simulation model but only for the most difficult interactions. Small separate models for each type of interaction were preferred over large integrated models, because separation of concerns is expected to result in models that are easier to create, maintain, translate and integrate into the overall simulation model. Each model can therefore be considered as a verified pattern for a specific type of interaction. Formal model checking of such basic patterns increases the confidence that the more complicated scenario simulation model will behave as expected.

Formal models are not necessarily updated as the scenario simulation model evolves because new models can easily be created whenever the need arises.

Figure 4.5 and 4.6 shows an example of a formal model of basic interactions and the verification results, respectively. This double-conformation pattern was used in several places in the overall scenario simulation model in the case study.

4.7.3 Modified framework

The SimJava2 simulation framework was chosen for the case study as described in section 4.4.5. It was modified and extended as described in chapter 4 in [3].

SimJava2 was modified to support easy modeling and analysis of the Cloud queue size as described in section 4.1 in [3]. These changes were made directly to the original SimJava2 classes. This solution is not optimal but it was chosen to avoid unnecessary complications such as creating an entirely new integrated simulation framework.

Furthermore, SimJava2 was extended with derived classes to facilitate simplified modeling as described in section 4.2 in [3]. To facilitate easy and efficient messaging, the classes `Sim_entity` and `Sim_event` were extended as shown in figure 4.7. These classes, `ExtSim_entitity` and `ExtSim_event`, are collected in a separate package, `/dtu/extsim`.

The changes and modifications only add convenience function to the SimJava2 simulation framework and do not change the fundamental architecture or behavior of the simulation engine.

4.7.4 Traditional modeling

The SimJava2 tutorial [23] describes how to use the framework to model systems. Figure 4.8 illustrates the major example from the tutorial. In this example, the `ProcessorSubsystem` class instantiates the entities (`Source`, `Processor`, `Disk1`, `Disk2`), links the ports, and starts the simulation in the `main(...)` function. The entity classes are extended from the `Sim_entity` class of the SimJava2 framework. The `Disk1` and `Disk2` are instantiated from the same entity class, `Disk`. The `body()`

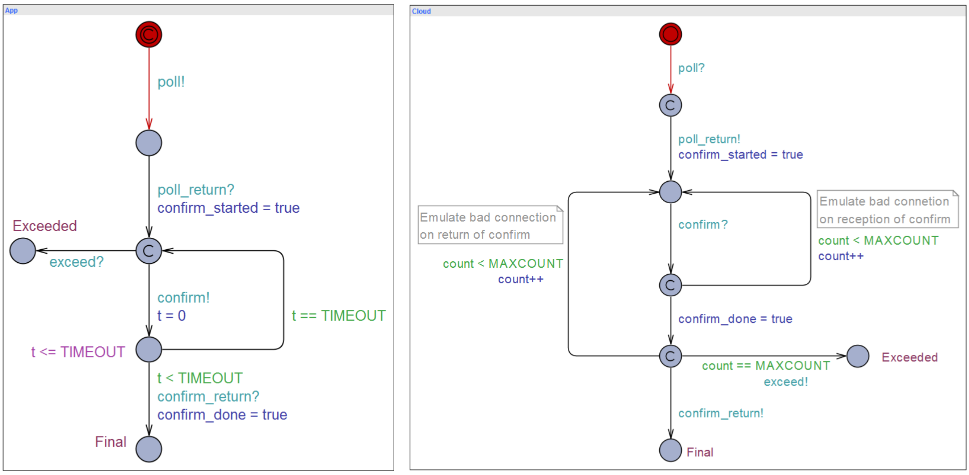
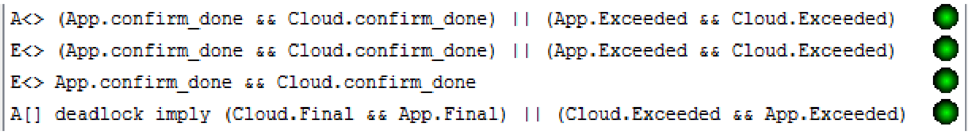


Figure 4.5: Formal model of basic interaction. This example shows a UPPAAL model of a double confirmation pattern that was used in several places in an overall scenario simulation model in the case study.



```
A<> (App.confirm_done && Cloud.confirm_done) || (App.Exceeded && Cloud.Exceeded)   ●
E<> (App.confirm_done && Cloud.confirm_done) || (App.Exceeded && Cloud.Exceeded)   ●
E<> App.confirm_done && Cloud.confirm_done                                          ●
A[] deadlock imply (Cloud.Final && App.Final) || (Cloud.Exceeded && App.Exceeded)  ●
```

Figure 4.6: Formal verification of the basic interaction that was modeled and shown in figure 4.5. The green circles shows that the queries passed the formal verification.

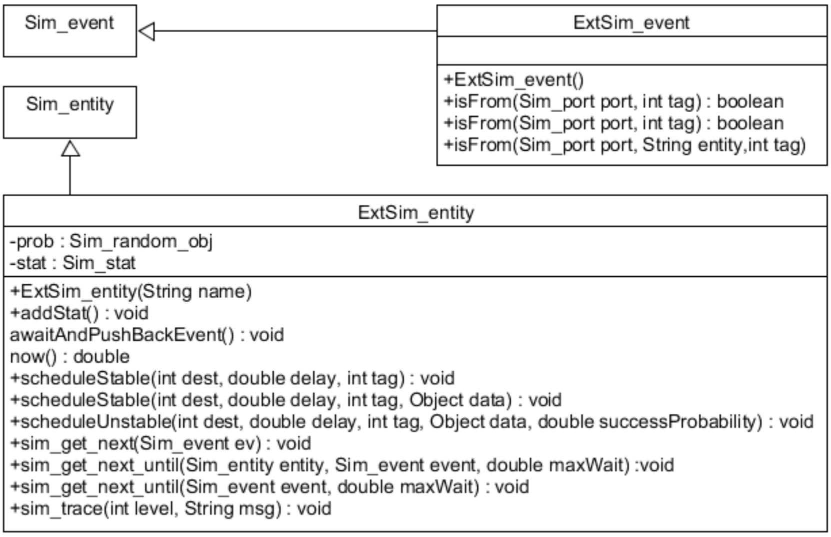
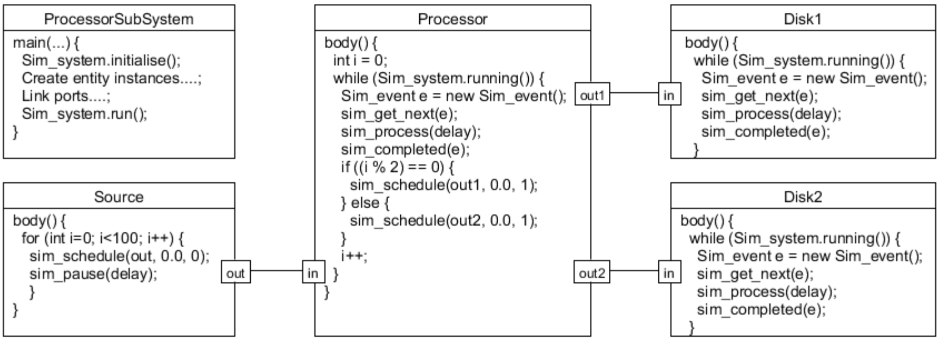

Sim_event
ExtSim_event
+ExtSim_event()
+isFrom(Sim_port port, int tag) : boolean
+isFrom(Sim_port port, int tag) : boolean
+isFrom(Sim_port port, String entity,int tag)
Sim_entity
ExtSim_entity
-prob : Sim_random_obj
-stat : Sim_stat
+ExtSim_entity(String name)
+addStat() : void
awaitAndPushBackEvent() : void
now() : double
+scheduleStable(int dest, double delay, int tag) : void
+scheduleStable(int dest, double delay, int tag, Object data) : void
+scheduleUnstable(int dest, double delay, int tag, Object data, double successProbability) : void
+sim_get_next(Sim_event ev) : void
+sim_get_next_until(Sim_entity entity, Sim_event event, double maxWait) :void
+sim_get_next_until(Sim_event event, double maxWait) : void
+sim_trace(int level, String msg) : void

Figure 4.7: ExtSim class diagram. The `ExtSim_entity` extends the `Sim_entity` with functions to facilitate communicating and tracing. The `ExtSim_event` extends the `Sim_event` with functions for easily identifying the sender and the type of message.



```
main(...) {
  Sim_system.initialise();
  Create entity instances....;
  Link ports....;
  Sim_system.run();
}
```

```
body() {
  for (int i=0; i<100; i++) {
    sim_schedule(out, 0.0, 0);
    sim_pause(delay);
  }
}
```

```
body() {
  int i = 0;
  while (Sim_system.running()) {
    Sim_event e = new Sim_event();
    sim_get_next(e);
    sim_process(delay);
    sim_completed(e);
    if ((i % 2) == 0) {
      sim_schedule(out1, 0.0, 1);
    } else {
      sim_schedule(out2, 0.0, 1);
    }
    i++;
  }
}
```

```
body() {
  while (Sim_system.running()) {
    Sim_event e = new Sim_event();
    sim_get_next(e);
    sim_process(delay);
    sim_completed(e);
  }
```

```
body() {
  while (Sim_system.running()) {
    Sim_event e = new Sim_event();
    sim_get_next(e);
    sim_process(delay);
    sim_completed(e);
  }
```

Figure 4.8: Model example from the SimJava2 tutorial [23]. The `Source` sends messages to the `Processor`, which upon reception of this message sends a message to either `Disk1` or `Disk2`. The `ProcessorSubsystem` sets up the model and starts the simulation.

functions of the entities are called by the SimJava2 framework when the simulation is started. From the figure, it is seen that the `Source` sends 100 messages separated by a random delay to the `Processor`. The `Processor` receives the messages from the `Source` and sends messages randomly to either `Disk1` or `Disk2`. These messages are finally received and processed by `Disk1` or `Disk2`. Notice that the communication between entity instances is carried out by sending and receiving messages through ports. The messages are passed from the output ports of the sending entities into central event queues in the `Sim_system` class. The messages are then forwarded to the input ports of the receiving entities. More details about the example is provided in the tutorial.

The use of ports to encapsulate the entities can be considered as a good modeling practice. Encapsulations make it is easy to reuse entities as in the example where the `Disk` entity class was reused for both `Disk1` and `Disk2`. However, using ports has a penalty in the form of increased modeling and simulation complexity, because ports and links and links must be resolved, whenever messages are transmitted. For models with extensive messaging, it will therefore be preferable to exchange messages between entities without using ports.

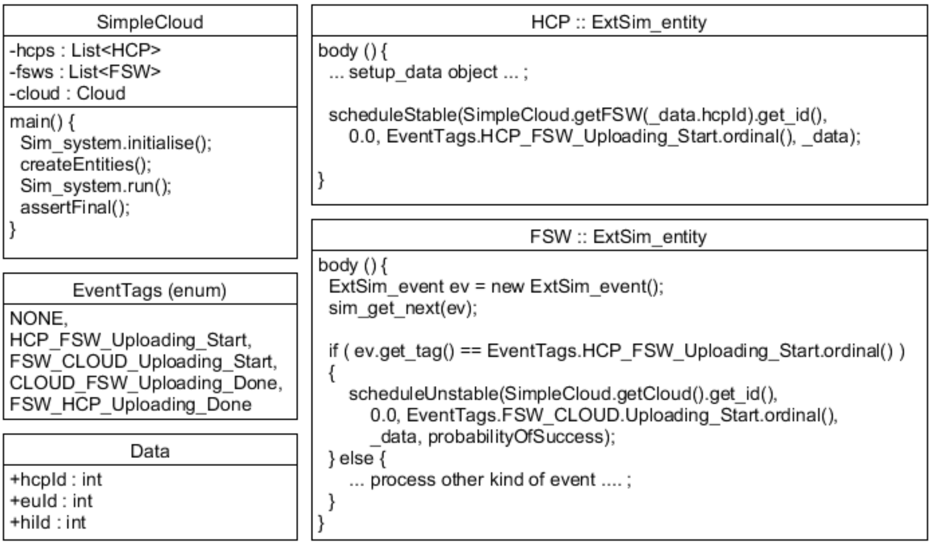
4.7.5 Avoiding ports

With SimJava2, it is possible to bypass the use of ports[3]. To accomplish this, the `Sim_entity.sim_schedule(dest, delay, tag, data)` function is utilized. Here `dest` is an integer that identifies the receiving entity. This identifier is obtained from the `get_id()` function of the receiving entity. The `delay` specifies the amount of delay between sending and receiving the message. The `tag` is an integer that identifies the type of message, which can be used by the receiving entity to determine what to do with the message. The `data` is an object (of any type) attached to the message.

4.7.6 Usage example

The `SimpleCloud` model included in section 3.2 in [3] can be used to illustrate the method of modeling with the extended simulation framework. Figure 4.9 shows a simplified class diagram that include partial descriptions of the implementations in the `main()` and `body()` functions.

[3] It will not be possible to send directly from entity to entity because the events must be queued such that the simulation framework can fetch and send the events when their timestamps match the simulation time. All messages are therefore stored in global event queues in the `Sim_system` class. However, the use of ports can be eliminated as described in the text.



```
-hcps : List<HCP>
-fsws : List<FSW>
-cloud : Cloud
```

```
main() {
  Sim_system.initialise();
  createEntities();
  Sim_system.run();
  assertFinal();
}
```

```
body () {
  ... setup_data object ... ;

  scheduleStable(SimpleCloud.getFSW(_data.hcpId).get_id(),
     0.0, EventTags.HCP_FSW_Uploading_Start.ordinal(), _data);

}
```

```
NONE,
HCP_FSW_Uploading_Start,
FSW_CLOUD_Uploading_Start,
CLOUD_FSW_Uploading_Done,
FSW_HCP_Uploading_Done
```

```
+hcpId : int
+euId : int
+hiId : int
```

```
body () {
  ExtSim_event ev = new ExtSim_event();
  sim_get_next(ev);

  if ( ev.get_tag() == EventTags.HCP_FSW_Uploading_Start.ordinal() )
  {
     scheduleUnstable(SimpleCloud.getCloud().get_id(),
        0.0, EventTags.FSW_CLOUD.Uploading_Start.ordinal(),
        _data, probabilityOfSuccess);
  } else {
     ... process other kind of event .... ;
  }
}
```

Figure 4.9: Minimal example of scenario model based on SimJava2 and ExtSim. Notice that the `Cloud` entity is not shown in the diagram.

The `SimpleCloud` main class contains an instance of the `Cloud` class and lists of `HCP` and `FSW` instances. The `main()` function initializes the simulation framework, creates the entity instances, starts the simulation, and asserts the final post-conditions after the simulation is stopped by the simulation framework.

The `HCP` class represents hearing care professionals and it extends the `ExtSim_entity` class. When the `body()` function is called at the beginning of the simulation, it sends a message to an associated `FSW` instance after setting up the correct values in the `_data` object. Here we want to model the message exchange on a stable connection because the `FSW` will always react to user inputs. This is done by using the `scheduleStable(...)` function (which wraps the previously mentioned `Sim_entity.sim_schedule(...)`) defined in the `ExtSim_entity` class. The message is tagged `HCP_FSW_Uploading_Start`.

The `FSW` entity represents the fitting software and it extends the `ExtSim_entity` class. When the `body()` function is called at the beginning of the simulation, it waits for the arrival of event messages. This is done by the calling `sim_get_next(...)` function defined in the `ExtSim_entity` class. If the tag of the received event message matches `HCP_FSW_Uploading_Start` then the `FSW` knows that it should start uploading `_data` to the cloud. In this case, the message from the `FSW` to the `Cloud` is modeled on an unstable connection, because there is some possibility that the message will be lost in transmission. This is done by using the `scheduleUnstable(...)` function defined in the `ExtSim_entity` class. If the event message received by the `FSW` entity matches any other event tags, e.g. `Cloud_FSW_Uploading_Done`, then those events are processed instead.

If the message is received by the `Cloud` entitity (not shown in the figure) and if the tag matches `FSW_CLOUD_Uploading_Start`, it will process the associated data object. A confirmation message tagged `CLOUD_FSW_Uploading_Done` will then be sent back to the `FSW` entity.

Notice that a global enumeration, `EventTags`, is used for keeping track of the `tags` used in the model. The names of the tags conform to the following format:

```
SOURCE_DESTINATION_CommandOrMessageName
```

With this naming scheme, it will be easy to get an overview of the messaging in the various parts of the model.

The minimal example that was presented above and shown in figure 4.9 does not describe the finer details of getting the model to work as intended. The `SimpleCloud` main class must provide functions to get access to the `HCP`, `FSW` or `Cloud` instances. If there are more than one pair of `HCP` and `FSW`, one must be able to identify the correct instance by associating identifiers of the relevant instances. Numerous initializations are performed but not shown for the `HCP`, `FSW` and `Cloud` entities. Many details of the `body()` functions have also been hidden in the figure. Finally, the example does not show how assertions are implemented. Such details are available in the code listings in section 3.2 in [3], however.

While the minimal example and the `SimpleCloud` example are acceptable for describing the basic methodology of modeling with SimJava2 and ExtSim, they are not sufficiently complex to evaluate the proposed approach to test-driven scenario modeling. The evaluation is therefore based on the test-driven modeling of a much more complex and large system as described in section 4.8.

4.8 Applying method on case

To evaluate the proposed method (section 4.6) and its implementations (section 4.7), the steps of the method were applied to the case study (section 4.5) as described in the following paragraphs[4].

[4] There is no correspondence between the modeled system and the system developed by GN Hearing, because it was decided by the company to isolate the modeling activities from the

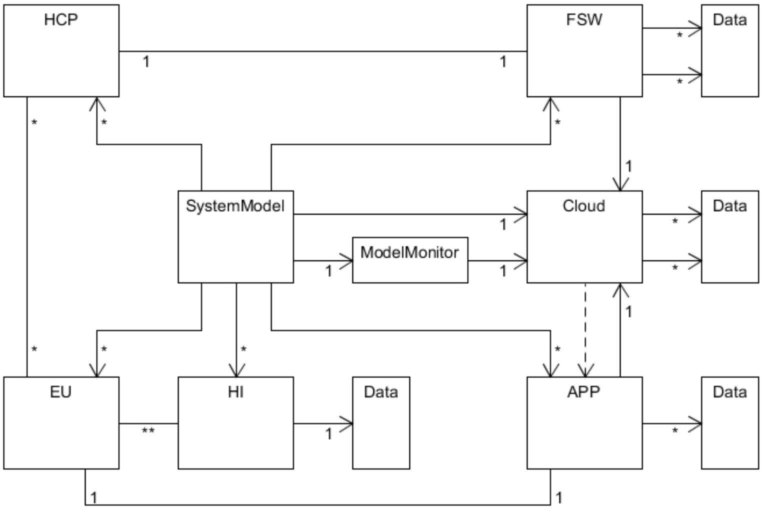


Figure 4.10: Simplified block diagram for simulation model (SysML BDD).

Step 1: A block diagram of the simulation model was created using SysML block definition diagram (BDD) notation as shown in figure 4.10. Notice that the associations are not used for communication between the classes but for identifying related entities, e.g. the FSW of the HCP. Communication is facilitated by sending messages as previously shown in figure 4.2. The SystemModel block represents the main class of the simulation model. The EU, APP, HI, HCP, FSW, and Cloud blocks represent the acting entities in the SUC. The ModelMonitor block is used for monitoring the state of the Cloud block during simulation to ensure that the simulation is terminated correctly. Data represents RFP and it contains a collection of status fields that are used for checking correct routing during and after the simulation. Each class and their associations are further described in chapter 5 in [3].

Step 2: SimJava2 was used as simulation framework. All acting entities (EU, APP, HI, HCP, FSW, Cloud, and ModelMonitor) were derived from the Sim_entity class. The SystemModel main class was derived from java.lang.object, and it included the main() function to initialize the simulation framework, to instantiate the subsystem classes, and to start the simulation. After termination of the simulation, the main() function continues by calling a function assertFinal()to assert the post-conditions.

activities of developing the actual system, such that interference from the case study did not cause delays in the extremely tight development schedule.

Step 3: The first action in the scenario (a) was added to the `body()` function of the `HCP`, such that a command was sent for each `EU` associated with the `HCP`. For the second step in the scenario (b), it was asserted that all `FSW` instances had uploaded the correct number of `Data` elements.

Step 4-6: The model was compiled and executed (step 4). No pre-conditions was added in this case study (step 5). The simulation was then started (step 6).

Step 7: The post-conditions failed as expected because the functionality of the scenario actions (a) and (b) was not yet defined in the `FSW` and `Cloud` entities.

Step 11: The missing functionality of the scenario actions (a) and (b) was then added to the `body()` functions of the `FSW` and `Cloud` entities.

Step 4-7: The model was initialized and executed again. No assertions failed after implementing the `body()` functions.

Step 8: New test cases were added to the model incrementally by iterating back to step 3, where new assertions representing post-conditions were added to the function `assertFinal()`. Assertions representing invariants were added to the `body()` functions of the involved entities. Several iterations of steps (3, 4, 5, 6, 7, 8, and 11) were completed to get an acceptable solution behaving according to the scenario from section 4.5.2.

Step A-C: Some parts of the modeling were notoriously difficult. Especially the loss of messages in the communication between various entities tended to cause inconsistent system states in early versions of the model. The optional sub-steps A-C were applied in such cases to provide robust patterns. A solid solution to the overall design problem required double-confirmation of several message exchanges due to the randomness of the scenario. Such patterns were therefore formally modeled and verified using the test-driven method described in chapter 3. Figure 4.5 shows an example of one of the double-confirm patterns that were used in the model (step A). Figure 4.6 shows four invariants expressed in the UPPAAL query language. The green dots show that the queries passed formal verification (step B). The UPPAAL models were manually converted to Java statements and incorporated into the simulation model (step C). Several such formally verified patterns were made during the development of the scenario models.

Further iterations of steps 3, 4, 5, 6, 7, 8 and 11 were executed to obtain a complete solution for the scenario. Test-driven modeling according to the proposed method was applied throughout the development of the scenario models. New functionality was added to model in small successive steps. We always adhered the rule of adding test case assertions, test case steps and finally system functionality in the sequence prescribed by the test-driven approach. Refactoring to clean up the model was applied several times during the modeling sessions. Simulation results often depended on the values of model parameters. Some values resulted in successful evaluation of assertions, while other values resulted in failed assertions or even fail to complete the simulations. To raise the level of trustworthiness, varying sets of parameters were simulated whenever major changes were made to the model.

Step 9: Numerous simulations with different parameter values were executed to obtain results for the mathematical forecasting that was used for predicting the functionality and performance of the system under realistic conditions. We focused on predicting the total number of events and the size of the event queues associated with the Cloud entity. We also focused on predicting the simulation time to investigate the feasibility of verifying the mathematical forecasting[5]. There are millions of acting entities (EUs, APPS, HIs, HCPs, FSWs, etc.) in realistic worst-case scenarios. However, the number of acting entities was limited to a maximum of approximately 200.000 for technical reasons and for limiting the required simulation time[6].

Simulation series: Several parameter constellations were simulated. First, a base simulation was run with a fixed set of parameters as described in section 2.1 and 2.2 in [4]. Then, three simulation series were run with varying parameters. The total number of RFPs was varied between 5.000 and 50.000 in one series. The total number of acting entities was varied between 20.102 and 200.102 in another series. The push probability was varied between 0 and 100 % in a third series. All simulations were conducted such that only one parameter was varied in each series. The other parameters were set to the same value as the base simulation. Further details concerning the parametric variation are provided in section 2.1 in [4].

Simulation results: Several data were logged during simulation. The impact of design decisions could therefore be analyzed and visualized. Examples of the results obtained from the simulations are discussed below.

[5] Different methods must be utilized to simulate with more than 200.000 to 300.000 acting entities because it will not be possible to create and run more threads. Such alternative methods are the discussed in section 4.9.7.

[6] Each entity is executed in a separate thread in SimJava2, which limits the number of entities to a little above 300.000 on the computer that was used for the simulations. Experiments showed that using more than approximately 200.000 caused significantly increased simulation time, which was probably due to significant growth in the overhead due to task swapping.

Processing time (PT) is a measure of the time it takes to process a RFP from it has been defined by the HCP until its final status has been set by the Cloud. Analyzing the simulation data showed that PT is largely independent of the number of acting entities and number of processed RFPs. However, lowering the push probability from 90 % to 60 % increased PT 3-4 times as shown in section 2.3 in [4].

The number of retries (NR) is a measure of how many times messages are re-send to counteract the loss of data during transmission. Analyzing the simulation data showed that no interaction required more than 6 retries. NR is largely independent of the number of acting entities and RFPs. Lowering the push probability changes the message patterns (decreasing the number push messages and increasing the number of notify messages) but does not change the distribution of NR significantly as shown in section 2.4 in [4].

The correlation of PT and NR is analyzed in section 2.5 in [4]. The correlation coefficient $\rho = -0,101$ calculated from the simulation data shows that there is virtually no correlation between PT and NR.

Step 10: To get indications concerning the feasibility of simulating scenarios with more entities and data packages, we wanted to estimate the simulation time (ST). To predict the data traffic into the cloud for such scenarios, we wanted to estimate the number of cloud events (CE) and the cloud event queue size (QS). Both CE and QS are essential parameters of the intended cloud solution.

The mathematical forecasting was based on data fitting and extrapolation techniques. Data fitting was used to identify a set of expressions that represented the data. It was assumed that each of these functions could be extrapolated beyond the data with acceptable accuracy.

Data fitting: We plotted ST, QE and QS versus the varied parameters for the simulation data obtained in step 9. All simulation data could be fitted with acceptable accuracy using the following generic function:

$$F(X) = F_0 \cdot \left\{1 + k \cdot (X - X_0)^q\right\} \tag{4.1}$$

where F represent ST, QS or CE. $F0$ is the value of the function for the base simulation. k is a scaling constant. X is the variable under consideration. X_0 is the value of the variable under consideration for the base simulation. And q is a quotient.

Table 4.3 shows 9 set of values for F_0, X_0, k and q that provide acceptable fit to the simulated data. Graphs of simulation data and the functional fits are shown in section 2.6 in [4].

Table 4.3: Data fitting constants used with equation (4.1).

	Simulation Time $F_0 = 00{:}03{:}52$	Events $F_0 = 107291$	Queue Size $F_0 = 481$
Packages $X_0 = 5000$	$k = 2.60 \cdot 10^{-6}$ $q = 1.6$	$k = 2.56 \cdot 10^{-6}$ $q = 1.6$	$k = 2.70 \cdot 10^{-2}$ $q = 0$
Entities $X_0 = 20101$	$k = 1.65 \cdot 10^{-9}$ $q = 2.1$	$k = 3.72 \cdot 10^{-5}$ $q = 1.0$	$k = 5.17 \cdot 10^{-5}$ $q = 1.0$
Push probability $X_0 = 90\%$	$k = -7.00 \cdot 10^{-2}$ $q = 1.0$	$k = -7.15 \cdot 10^{-2}$ $q = 1.0$	$k = -9.90 \cdot 10^{-2}$ $q = 1.0$

Estimating: If the variables are independent, then each estimator function can be separated into a product of functions that only depends on one variable. Using the notation $\widetilde{F}_n = F_n/F_0$, we get the following generic function:

$$F(X, Y, Z) = F_0 \cdot \widetilde{F}_1(X) \cdot \widetilde{F}_2(Y) \cdot \widetilde{F}_3(Z) \tag{4.2}$$

The specific estimator functions were derived from equation (4.1) and (4.2) using the constants from table 4.3. The resulting functions are presented in section 2.7 in [4]. ST, CE and QS were then calculated for seven different scenarios that were later simulated to investigate the accuracy of the estimator functions as described below.

Verifying estimates: The results of the verifications are shown in table 4.4. From the table it is seen that (1) the simulated values of ST, CE, and QS deviated up to +102%, -50%, and +6.4% from the estimates, respectively; and (2) the parameters of the scenarios were far from realistic scenarios but rather close to the parameters used in the simulation series from which the estimates are calculated. These results clearly show that estimates should always be verified before they are used in forecasts, as done here. The root causes of the deviations concerning ST and CE were not investigated further, but hidden dependencies between the parameters were suspected of partially invalidating the approach of separating the parameters in this particular case.

Using estimates: The estimates of ST and QE are somewhat imprecise and the deviations may grow further for parameters representing different scenarios. However, lacking other means for predicting performance, the estimator functions were used for estimating a large but realistic scenario. The estimates for processing X millions RFPs (confidential) in a realistic system (Y millions EUs (confidential), 1.000 HCPs, 90% push probability, 2 HIs and 1 APP per EU, and 1 FSW per HCP) are:

$$\text{ST} > 9.000 \text{ years} \qquad \text{QE} > 1.6 \cdot 10^{11} \qquad \text{QS} \sim 100.000$$

These values clearly show that it is virtually impossible to simulate such scenarios.

Table 4.4: Verification of the estimator functions.

Check nr.:	#1	#2	#3	#4	#5	#6	#7
				Parameters:			
HCPS	50	60	70	80	100	120	75
RUNS	500	100	100	100	100	100	175
EUS / HCP	200	100	100	100	100	100	175
HIS / EU	2	2	2	2	2	2	2
Push Prob.	90	90	90	90	90	90	95
Entities	40.101	24.121	28.141	32.161	40.201	48.241	52.651
Packages	25.000	6.000	7.000	8.000	10.000	12.000	13.125
				Simulation time:			
Simulated	07:31:02	00:07:03	00:11:19	00:19:01	00:37:35	01:03:16	01:31:51
Predicted	03:43:19	00:04:47	00:07:18	00:12:11	00:34:06	01:24:09	01:24:53
Deviation	102,0%	47,4%	55,0%	56,1%	10,2%	-24,8%	8,2%
				Number of cloud events:			
Simulated	4.226.190	147.685	193.420	255.968	377.847	511.053	563.999
Predicted	3.834.636	143.258	207.628	301.023	585.280	1.017.667	885.404
Deviation	10,2%	3,1%	-6,8%	-15,0%	-35,4%	-49,8%	-36,3%
				Cloud queue size:			
Simulated	1.041	578	672	809	988	1.138	655
Predicted	978	581	681	781	981	1.181	652
Deviation	6,4%	-0,5%	-1,3%	3,6%	0,7%	-3,6%	0,5%

4.9 Discussion

4.9.1 Results

The proposed method was applied to the modeling and verification of a large and complex smart system. The modeled system contains 6 kinds of entities (EUs, HCPs, APPs, HIs, FSWs and the Cloud) with an expected total of up to 4·X million instances (confidential).

Knowledge about the intended system was collected into a descriptive SysML model. SysML was also used for creating a block diagram to provide an overview of the scenario simulation model. The simulation model was developed in small steps according to the proposed test-driven approach. The SimJava2 simulation framework was modified and extended to facilitate easy modeling that allows communication between entities without using ports.

The model has been simulated with different combinations of parameters. Simulations were executed in series where only one parameter was varied. In one series, the model was simulated with 5.000 to 50.000 processed RFT packages. In another series, the model was simulated with 5.000 to 200.102 acting entities. In a third series, the model was simulated with a push probability from 0% to 100%. Based on the simulations we could conclude that the processing times for the RFT packages are largely independent of the number of entities and packages. The push probability, however, affects both the average processing time and the number of re-transmissions that were required to compensate for lost message.

The simulation results were subjected to further analysis. Data fitting of simulation results and extrapolating beyond the data allowed us to forecast mathematically the required simulation time, number of cloud events and size of the cloud event queue for other combinations of parameters. To verify the estimates provided by the mathematical forecasting, additional series of simulations were executed. The estimates of simulation time were generally too optimistic and required up to 100% more than predicted. The estimates of the number of cloud events were found to be somewhat too pessimistic. The actual number of events were only 50% of the predicted value in one case. The estimates of the required cloud queue size were found to be more precise. The maximum deviation between predicted and simulated queue sizes was less than 7%.

The goals of the case study were outlined in section 4.5.4. The goal of evaluating the proposed method has been achieved by successfully conducting the activities described in section 4.8 even though it was shown that the part concerning mathematical forecasting is somewhat problematic because it may lead to imprecise results.

The goal of verifying the applicability of the chosen modeling technologies has been achieved successfully even though the choice of SimJava2 poses some limitations as described in section 4.9.6. The results obtained from the case study indicates that the proposed method is useful for developing viable solutions to the design problem.

The detailed goals for developing viable solutions were generally achieved. Absence of potential deadlocks cannot be guaranteed by executing simulations without guaranteeing that all states and parts of the model have been exercised. However, the absence of deadlocks was verified by formal model checking for basic interaction patterns, which increases the trustworthiness of the overall simulation model. No deadlocks were observed in the final versions of the simulation models. The goals of verifying that stored data always have the correct values and that messages, events, and data are routed correctly were achieved, by passing all assertions relating to these requirements. The goal of estimating cloud loading (number of events, queue size, etc.) was achieved by mathematical forecasting based on data fitting and extrapolation techniques. However, the estimates cannot be verified as described in section 4.9.5. The goal of exploring the design space was achieved by simulating the model with different parameter values. Feasible ranges for sensitive parameters were identified during such experiments.

4.9.2 Observations

During the modeling sessions, it was observed that:

(1) Using a programming language (Java) for the simulation model caused a tendency to change the focus from systems thinking to programming.

(2) Comprehending and maintaining an overview of exchanging messages in complex simulation models can be difficult, which occasionally makes it challenging to create good solutions to the design problem.

(3) Defining stop conditions can be very tricky, because the simulation must stop eventually but not before all data and entities have reached the states that allow the post-conditions to pass.

For issue (1), it was found easier to make a few formal models in UPPAAL to solve specific problems and later convert these models to scenario simulation models. Systems thinking and solving programming problems could therefore be separated into separate phases.

Issue (2) was addressed by creating SysML sequence diagrams of selected parts of the modeled scenario, SysML internal block diagrams showing message exchanges, and call graphs showing the functional hierarchies. The UPPAAL models also helped to provide a better overview.

Issue (3) may appear to be trivial, but it turned out to be somewhat difficult to define proper stop conditions. For the case study, this issue was addressed partly by adding a `ModelMonitor` entity that checked the state of all data stored within the `Cloud` at regular intervals. The effort and insight required to define proper stop conditions should not be underestimated.

Using the test-driven approach with such mitigations resulted in a process that was relatively easy to execute. The cognitive load during modeling was perceived as low, even though it was not quantified in this project.

4.9.3 Learning points

Important learning points from conducting the case study are:

- Useful information is hard to get from stakeholders or existing documentation:
  - Much time was spent on identifying relevant use cases and scenarios.
  - It was difficult to get accurate information concerning the desired steps and interactions of the scenario under consideration.
  - Test cases based on well-defined requirements are needed for using test-driven modeling methods successfully. Clarifying and converting requirements into usable test cases required much effort.
- Textual modeling using programming languages tend to cause issues that prevent developing the best solution. Graphical modeling with SysML and UPPAAL was useful for maintaining an overview of the problem and for obtaining well-designed solutions.
- The model is easily broken when large or many changes are made. Repairing models to pass assertions can be very time consuming. It was found better to make changes in very small increments and run simulations between each change.
- Large scenarios require much time to simulate. It was found better to test with small scenarios during the development of the model. To ensure that the model can pass assertions in all cases, larger scenarios must be simulated regularly, though.
- Manual data fitting is somewhat cumbersome and error prone. A more automated process using dedicated tools is desirable.

- Estimator functions for mathematical forecasting can be very imprecise (probably due to hidden dependencies between parameters) and should be checked by comparing estimates and simulations results for various scenarios.

Otherwise, the proposed method was found to be easy to apply to the case study.

4.9.4 Advantages

The proposed method has several advantages:

1. It allows test-driven modeling and verification of very large and complex systems, which is difficult or impossible using previous methods.
2. Simulating all parts of the model concurrently exposes unintended emergent behavior, which is not exposed or captured by e.g. unit testing.
3. Using formal model checking of basic interactions increases the trustworthiness of the models.
4. Mathematical forecasting based on data fitting and extrapolating allow analysis of large scenarios that cannot be simulated or otherwise analyzed.
5. It utilized only existing tools which may be substituted independently by better tools in the future without changing the methodology.
6. Programmers with experience in TDD and Java can easily learn the method.
7. Existing software source code can easily be included in the models, if it is written in the same or compatible programming languages (e.g. Java source code in SimJava2 models, or assembler source code in SystemC models).

The proposed method is therefore considered a major contribution to the field of test-driven modeling of CECPS systems.

4.9.5 Disadvantages

The proposed method has the following disadvantages:

1. Many simulations are required with parameter co-variation to estimate large-scale scenarios.

2. Mathematical forecasting is difficult, potentially imprecise and cannot be thoroughly checked due to infeasible simulation times.
3. Programming skills are required for creating simulation models, and knowledge of formal modeling is required if the optional steps are utilized.

Most severe is the disadvantage of potentially imprecise mathematical forecasting. Moderately imprecise estimates are likely to be very useful in the early phases of engineering the system under consideration. The knowledge of the system may be somewhat limited and the models may be the result of much abstraction and simplification. Moderate precision may therefore be sufficient at this stage of development. Extremely imprecise estimates that deviate orders of magnitudes from correct values are useless, because it is impossible to conclude anything from the forecasts. Contrarily, such estimates may lead to wrong conclusions, if the uncertainty is ignored. Guesstimating may provide results that are more reliable in such extreme cases. Between these cases, there is a range where the imprecision may be large but not extreme. In such cases, the estimates may provide knowledge of system properties, which may otherwise be very difficult to get. Here, the estimates may be of strong value even though the precision is limited. Future work should explore other simulation frameworks that might allow simulating larger scenarios and reducing the need for mathematical forecasting.

4.9.6 Limitations

The proposed method and its implementation have some limitations:

1. SimJava2 uses one thread per entity, which limits the number of entities to approximately 200.000 on the computer used in the case study[7].
2. Simulating with relevant parameters may be impractical, because varying parameters simultaneously may result in extremely long simulation times[8].
3. Modeling and verification of user interfaces are not included in the method.
4. Unit testing of individual entities is not included in the method.

[7] Computer: 16 GHz RAM and Intel i7-3630QM CPU @ 2.40GHz.

[8] In the case study, scenarios with 200.000 entities and 5.000 packages could be simulated in less than 12 hours, and scenarios with 20.000 entities and 50.000 packages could be simulated in less than 5 hours. However, simulating with 200.000 entities *and* 50.000 packages was estimated to take from 10 days to 3 months depending on the value of the push probability. Realistic scenarios with 4·X million entities (confidential) and Y million packages (confidential) was estimated to require more than 9.000 years!

5. The case study only addressed discrete event simulations, while cyber-physical systems often necessitate heterogeneous models that require several modes of computation during simulation.

These limitations will not be relevant for all potential users of the method, but they should nevertheless be addressed in future work.

4.9.7 Improvements

The disadvantages and limitations of the proposed method can partly be addressed by introducing improvements to the implementation of the proposed method. Reduction of simulation time is of major importance because:

- It will allow simulating larger scenarios and reduce the need for potentially imprecise mathematical forecasting.
- It will reduce the time needed for designing and analyzing systems and thus increase engineering efficiency.

Several methods for reducing simulation time are possible:

1. Use super computers with massive parallelism.
2. Use distributed computing.
3. Use more efficient simulation frameworks.
4. Use SimJava2 with brokers to avoid massive generation of threads.
5. Create new simulation engine.

These proposals are further discussed in section 2.2 in [3].

4.9.8 Future work

Eliminating the need for mathematical forecasting should be prioritized in the future work due to its limitations. However, significantly faster simulators are needed for this. The case study indicated that the required simulation time for a realistic scenario would exceed 9.000 years. The simulator should therefore be able to execute at least

1.000.000 times faster to allow simulation of such scenarios. Several breakthroughs are needed to achieve this. Future work should therefore be devoted to the following:

- Substituting SimJava2 with a thread-lean simulation framework such as the core of CloudSim to reduce simulation time.
- Incorporating distributed computing in the simulation framework to reduce simulation time further.
- Using super computers with massive parallel computing for fast distribution of simulation tasks.
- Incorporating multi-level simulation techniques [48] in the simulation framework to reduce simulation time further.

Finally, more cases should be modeled to verify the method further.

4.9.9 Conclusion

A method for test-driven modeling of smart systems has been proposed and described. The method utilizes a combination of descriptive models (SysML), formal models (UPPAAL), scenario simulation models (SimJava2), and mathematical forecasting based on data fitting and extrapolation techniques. Other tools may substitute the chosen tools without changing the method.

By using the proposed method, it was possible to analyze and design scenarios for transmitting and processing data reliably in a cloud-enabled hearing system. Basic interactions were verified by using formal model checking with the UPPAAL tool. SimJava2 was used for creating a simulation model of the overall scenario. The simulation model was executed in several series with varying values for parameters of interests. Fitting the obtained simulation data with mathematical functions allowed us to predict system qualities for scenarios that were too large to simulate.

The modeling and simulating provided insight into the critical parameters and their impact on system behavior and performance. We were able to quantify the impact, and define acceptable ranges for the values of these parameters.

Future work is needed to overcome some difficulties of the proposed method. Identifying and using more efficient simulation frameworks are required to allow simulation of larger systems and to reduce simulation time. Several breakthroughs are needed to create feasible simulators, though. Until such simulators become available, the proposed method provides a reasonable alternative to analyze CECPS systems with millions of users.

4.10 References

[1] Allan Munck
"Hearing systems."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[2] Allan Munck
"Model-Based Systems Engineering Guidelines."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[3] Allan Munck
"Smart systems modeling."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[4] Allan Munck
"Smart systems simulation code and results."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[5] Allan Munck and Jan Madsen.
"Test-driven modeling and development of cloud-enabled cyber-physical smart systems".
Accepted for presentation at 11th Annual IEEE International Systems Conference, 2017. To be published in the conference proceedings in IEEE IEL and IEEE Xplore.

[6] Scott W. Ambler.
"Agile model driven development is good enough."
IEEE Software 20.5 (2003): 71-73.

[7] Scott W. Ambler.
"Agile Model Driven Development (AMDD)."
XOOTIC Symposium 2006, (p.13). 2006.

[8] David S. Janzen and H. Saiedian.
"Does test-driven development really improve software design quality?"
Software, IEEE 25.2 (2008): 77-84.

[9] Forrrest Shull et al.
"What do we know about test-driven development?"
IEEE software 27.6 (2010): 16-19.

[10] Wikipedia.
"Smart system."

Revision: 14:59, 5 January 2015,
https://en.wikipedia.org/wiki/Smart_system.

[11] Diane Cook and Sajal Das.
"Smart environments: Technology, protocols and applications."
Vol. 43. John Wiley & Sons, 2004.

[12] Max Mühlhäuser.
"Smart products: An introduction."
Constructing ambient intelligence. Springer Berlin Heidelberg, 2007. 158-164.

[13] Georges Akhras.
"Smart materials and smart systems for the future."
Canadian Military Journal 1.3 (2000): 25-31.

[14] Accellera - Systems Initiative.
"SystemC."
http://www.accellera.org/downloads/standards/systemc

[15] IEEE Standards Association.
"IEEE Standard for Standard SystemC ® Language Reference Manual."
9 January 2012, IEEE Std 1666^{TM} - 2011 (Revision of IEEE Std 1666-2005)
http://standards.ieee.org/getieee/1666/download/1666-2011.pdf

[16] Peter Gorm Larsen, Kenneth Lausdahl, and Nick Battle.
"VDM-10 Language Manual."
kurser.iha.dk/eit/tivdm2/VDM10_lang_man.pdf

[17] The University of Edinburgh - School of Informatics
"SimJava - Website."
http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/

[18] Mark Little.
"JavaSim simulation classes and examples."
The University of Newcastle upon Tyne, Department of Computing Science, Computing Laboratory.
https://github.com/nmcl/JavaSim

[19] Mark Little.
"Discrete event simulation in C++."
The University of Newcastle upon Tyne, Department of Computing Science, Computing Laboratory.
https://github.com/nmcl/C--SIM

[20] Univeristy of Georgia - Computer Science Department.
"JSIM: A Java-Based Simulation and Animation Environment."
http://cobweb.cs.uga.edu/~jam/jsim/

[21] The University of Edinburgh - School of Informatics.
"SimJava - Download - Version 2.0."
`http://www.inf.ed.ac.uk/research/isdd/admin/package?download=62`

[22] The University of Edinburgh - School of Informatics.
"SimJava v2.0 API Specification."
`http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/doc/index.html`

[23] The University of Edinburgh - School of Informatics.
"SimJava - Tutorial."
`http://www.dcs.ed.ac.uk/home/simjava/tutorial/`

[24] G. Brambilla, M. Picone, S. Cirani, M. Amoretti and F. Zanichelli.
"A simulation platform for large-scale internet of things scenarios in urban environments."
Proceedings of the First International Conference on IoT in Urban Space (pp. 50-55). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[25] L. Zhang, X. Ma, J. Lu, T. Xie, N. Tillmann and P. De Halleux.
"Environmental modeling for automated cloud application testing."
IEEE software 29.2 (2012): 30-35.

[26] L. Zhang, T. Xie, N. Tillmann, P. De Halleux, X. Ma and J. Lv.
"Environment modeling for automated testing of cloud applications."
IEEE Software, Special Issue on Software Engineering for Cloud Computing 1 (2012).

[27] Markus Miche, K. Baumann, J. Golenzer and M. Brogle.
"A simulation model for evaluating distributed storage services for smart product systems."
International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services, (pp. 162-173). Springer Berlin Heidelberg, 2011.

[28] Frank Dabek, B. Zhao, P. Druschel, J. Kubiatowicz and I. Stoica.
"Towards a common API for structured peer-to-peer overlays."
International Workshop on Peer-To-Peer Systems (pp. 33-44). Springer Berlin Heidelberg, 2003.

[29] Alberto Medina, A. Lakhina, I. Matta and J. Byers.
"BRITE: An approach to universal topology generation."
Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings (pp. 346-353). Ninth International Symposium on. IEEE, 2001.

[30] Hitesh Marwaha.
"A Comprehensive Review of Cloud Computing Simulators."
Journal of Information Sciences and Computing Technologies 4.1 (2015): 281-286.

[31] Mr. Manjunatha s, Bhanu Prakasha and H. M. Balakrishna.
"A Detailed Survey on various Cloud computing Simulators"
International Journal of Engineering Research, Volume No.5 Issue: Special 4, pp: 790-991, 20 May 2016.

[32] Pericherla S. Suryateja.
"A Comparative Analysis of Cloud Simulators."
International Journal of Modern Education and Computer Science (IJMECS) 8.4 (2016): 64.

[33] M. A. Kaleem and P. M. Khan.
"Commonly used simulation tools for cloud computing research."
Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on. IEEE, 2015.

[34] Utkal Sinha and Mayank Shekhar.
"Comparison of Various Cloud Simulation tools available in Cloud Computing."
International Journal of Advanced Research in Computer and Communication Engineering 4.3 (2015).

[35] Kalpana Ettikyala and Y. Rama Devi.
"A Study on Cloud Simulation Tools."
International Journal of Computer Applications 115.14 (2015).

[36] Arif Ahmed and Abadhan Saumya Sabyasachi.
"Cloud computing simulators: A detailed survey and future direction." Advance Computing Conference (IACC), 2014 IEEE International. IEEE, 2014.

[37] Azin Oujani and R. Jain.
"A Survey on Cloud Computing Simulations and Cloud Testing."
`https://www.cs.wustl.edu/~jain/cse567-13/ftp/cloud.pdf`.

[38] Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros.
"Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities."
High Performance Computing & Simulation, 2009. HPCS'09. International Conference on. IEEE, 2009.

[39] Rodrigo N. Calheiros et al.
"Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services."
arXiv preprint arXiv:0903.2525 (2009).

[40] Rodrigo N. Calheiros et al.
"CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms."
Software: Practice and Experience 41.1 (2011): 23-50.

[41] Rodrigo N. Calheiros et al.
"EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications."
Software: Practice and Experience 43.5 (2013): 595-612.

[42] Antonio Navarro Perez and Bernhard Rumpe.
"Modeling cloud architectures as interactive systems."
arXiv preprint arXiv:1408.5705 (2014).

[43] Faruk Caglar et al.
"Transitioning to the cloud?: a model-driven analysis and automated deployment capability for cloud services."
Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and CLoud computing. ACM, 2012.

[44] Alexander Bergmayr et al.
"UML-based Cloud Application Modeling with Libraries, Profiles, and Templates."
CloudMDE@ MoDELS. 2014. (pp. 56-65).

[45] Martin Fleck et al.
"Towards Pattern-Based Optimization of Cloud Applications."
CloudMDE@ MoDELS. 2014. (pp. 16-25).

[46] Vilen Looga et al.
"Mammoth: A massive-scale emulation platform for internet of things."
2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems. Vol. 3. IEEE, 2012.

[47] Vilen Looga.
"Energy-awareness in large-scale internet of things networks."
Proceedings of the 2014 workshop on PhD forum. ACM, 2014.

[48] Gabriele D'Angelo, Stefano Ferretti, and Vittorio Ghini.
"Simulation of the Internet of Things."
To appear in Proceedings of the IEEE 2016 International Conference on High Performance Computing and Simulation (HPCS 2016).

[49] Pradeeban Kathiravelu and Luis Veiga.
"Concurrent and distributed cloudsim simulations."
2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems. IEEE, 2014.

[50] Pradeeban Kathiravelu and Luis Veiga.
"An adaptive distributed simulator for cloud and mapreduce algorithms and architectures."
Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on. IEEE, 2014.

[51] Cristyan Manta-Caro and Juan M. Fernández-Luna.
"A discrete-event simulator for the web of things from an information retrieval perspective."
2014 IEEE Latin-America Conference on Communications (LATINCOM). IEEE, 2014.

[52] Bhathiya Wickremasinghe, Rodrigo N. Calheiros and Rajkumar Buyya.
"Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications."
2010 24th IEEE International Conference on Advanced Information Networking and Applications. IEEE, 2010.

[53] Gerd Behrmann, Alexandre David and Kim G. Larsen.
"A Tutorial on Uppaal 4.0 - Updated November 28, 2006"
`http://www.uppaal.com/admin/anvandarfiler/filer/uppaal-tutorial.pdf`.

Chapter 5

Discussion

The topics of the project and this thesis were introduced in chapter 1. The project was executed in the context of hearing systems and it was conducted at the company GN Hearing. Methods for selecting MBSE tools were discussed in chapter 2 and modeling guides for MBSE are presented in section 2 in [1]. Methods for test-driven modeling of embedded systems and smart systems were discussed in chapter 3 and 4, respectively. Guidelines for using all proposed methods are presented in section 3 in [1].

This chapter discusses and summarizes the overall research. The overall results and answers to the research questions are presented in section 5.1 and the evaluations of the research hypotheses are presented in section 5.2. Section 5.3 presents observations that were registered during the research. Problems and open issues are discussed in section 5.4 and 5.5, respectively. Future work is discussed in section 5.6. The potential impact of the research is discussed in section 5.7. Section 5.8 presents the overall conclusion of the thesis. References are finally listed in section 5.9.

5.1 Results and research answers

This section summarizes the research answers (RA) concerning the research questions (RQ) that were defined in section 1.5. The main research questions, RQ 1 and RQ

2, are answered by answering the derived questions. The answers to the questions derived from RQ1 are discussed in section 5.1.1. The answer to the questions derived from RQ2 are discussed in section 5.1.2.

5.1.1 MBSE tool selection

The research questions concerning selection of MBSE tools were addressed in chapter 2 and they can be answered accordingly:

RQ 1.1 *Which methods have previously been applied to selecting, implementing and using systems engineering tools?*

RA 1.1 Literature search has revealed several existing methods for selection of vendors, requirements management tools, modeling tools, simulation tools, software packages, etc. Most selection methods only concern comparative ratings of tools based on evaluations of relevant selection criteria. Some practitioners use qualitative methods where features, functionality and performance of tools are evaluated and compared without scoring the alternatives. Other practitioners use quantitative methods where the requirements are evaluated for each tool, such that a ranking of the tools can be calculated by using WSM, AHP or similar methods. Some researchers, e.g. Gotel and Mäder [14][15], also consider other aspects of selecting, implementing and using tools to provide guides for obtaining a wider solution that include tools, processes, people, etc.

RQ 1.2 *Which limitations or disadvantages of existing methods for selecting and implementing systems engineering tools necessitate modified or new methods?*

RA 1.2 Previous methods for tool selection concern very specific domains such as tools for requirements management, modeling and simulation. Comprehensive methods for general selection of systems engineering tools have not been found during the research. No selection criteria exists for selecting general systems engineering tools. New methods for selecting of such tools are therefore needed. The proposed method include steps to obtain requirements and selection criteria for all types of systems engineering tools.

Known methods only concerned single-tool scenarios with simple feature evaluation or ranking of tools. No methods addressed the multi-tool situation where several tools are needed to meet all essential needs. New methods to accommodate the multi-tool scenarios are therefore needed. The proposed method includes steps for both single-tool and multi-tool scenarios.

No known methods used any form of verifying and validating the choice of tools. However, it is essential to verify the tool solution and know exactly which requirements the tools fulfill and which requirements are dissatisfied. Similarly, it is essential to validate the tool solution and identify requirements that may otherwise be overlooked or ignored in the selection process. New methods that include verification and validation of the selected tool solution are therefore needed. The proposed method includes such steps as described below.

RQ 1.3 *How can tool solutions be verified and validated before they are implemented and used on a daily basis?*

RA 1.3 Tool solutions can be verified by analyzing the gaps between tool requirements and tool capabilities. It is therefore possible to identify the requirements that cannot be satisfied. No previous method includes such gap analyses.

Literature review has shown that most selection methods rely on generic selection criteria that have not be validated. More reliable selection criteria are obtained by the methods that utilize surveys or interviews of stakeholders to obtain requirements. However, there are risks of overlooking important requirements and of stating superfluous requirements (gold-plating). Combining interviews with observations of stakeholders performing their tasks related to systems engineering provide the most reliable method for obtaining requirements for a tool solution that can be validated.

RQ 1.4 *What can be achieved by using existing or newly proposed methods for selecting and implementing tools?*

RA 1.4 The author (me) has observed several cases where systems engineering tools were selected after a brief unstructured evaluation of few tools that had been recommended by persons that were trusted as reliable sources. In such cases, the required functionality cannot be guaranteed, though. The use of such unstructured methods may therefore cause limited scope and reduced usefulness of the acquired tool setup.

By using a structured approach, the analyst can ensure that relevant requirements are included in the analyses. Systematic searching for tools reduces the likelihood of overlooking good candidates. Thoroughly evaluating tools and correlating the tool capabilities to the actual needs increase the likelihood of obtaining the best possible candidates. Inclusion of stakeholders in the selection process reduces the risk of deliberate or unintended rejections of the chosen solutions.

RQ 1.5 *What are the advantages of the proposed method for selecting, implementing and using tool solutions?*

RA 1.5 The proposed method outlines four major steps for justifying and initiating the change project, developing tool specification and evaluation criteria, investigating tools and selecting best candidates, and implementing, maintaining and evolving the chosen solution. All relevant phases in the process of changing from documents to models are therefore considered in the proposed method.

Obtaining requirements for MBSE tools is particularly difficult due to the differences of the various activities in systems engineering. The classifications of system characteristics and modeling disciplines allow the analyst to ask specific questions to extract the relevant requirements from the stakeholders. The proposed method includes further steps for analyzing and specifying requirements to ensure that MBSE tools are evaluated on relevant criteria.

In many cases, it will not be possible to identify a single tool that meets all requirements. However, no prior method acknowledges or addresses this problem. The proposed method supports both single-tool and multi-tool scenarios. The use of set theory and simple mathematical operations allow the analyst to determine feasible sets of tools that in combination meets all essential requirements. The best set of tools can be chosen subsequently by ranking the feasible sets and select the set with highest score.

The proposed method includes gap analyses not only to verify the chosen solution but also to assist in the selection of promising candidates. This analysis can be performed without implementing and using the tools on daily basis. The gaps are analyzed for each category and priority-level of requirements to facilitate easy overview of the results. Therefore, non-conformance to e.g. required, desired and optional features can be demonstrated easily.

The proposed method uses interviews to obtain requirements with the optional possibility of using observational studies of stakeholders performing work tasks that are relevant for the tool solution. This decreases the likelihood of overlooking essential requirements and increases the likelihood of avoiding superfluous requirements compared to methods that use generic selection criteria. Having the correct requirements increases the likelihood of selecting tool solutions that can be validated.

RQ 1.6 *What are the disadvantages and limitations of the proposed method for selecting, implementing and using tool solutions.*

RA 1.6 Observing as a method for obtaining tool requirements is only optional in the proposed method. Basing the analyses on interviews is more likely to cause over-specification because stakeholders have a tendency to ask for more than they really need. Essential requirements may also be overlooked by relying on interviews. However, the effects of using observations rather than interviews cannot be derived from this research, because stakeholder observations have not been used in the case studies. Such research is left for future work.

The proposed method is highly systematic and requires much dedication and work to use. It therefore requires much time and consumes many resources that may not always be available. Using a thorough method in the selection process may also uncover other issues concerning work methods and processes. This may lead to other costly change projects that could otherwise be avoided, because premature solving of such issues may be unnecessary. A comprehensive method may thus abstain some organizations to undertake the work and use easier but less reliable methods.

The research thus uncovered scenarios that have not been sufficiently considered in previous work. A comprehensive method with novel elements was therefore developed and subsequently tested on two case studies with good results. Potential improvements and future work have been identified and proposed.

5.1.2 Test-driven modeling

The research concerning test-driven modeling of embedded and smart systems is described in chapter 3 and 4. The research questions can be answered accordingly:

RQ 2.1 *How have test-driven methods previously been applied to systems modeling?*

RA 2.1 Literature search has revealed several existing methods for test-driven modeling of systems or system parts. Zhang [2] described a method for test-driven software modeling utilizing simulations of message sequence charts (MSC). Hayashi et al. [3] described a method and developed a simple tool for test-driven UML modeling of software systems that utilize simulations of unit and scenario tests. Luna et al. [4][5] described test-driven, model-based web engineering with a focus on usability of web applications. Zhang and Patel [6] described an agile model-driven development approach for software development utilizing UML sequence diagrams for tests and UML state machines for system models. Zugal et al. [7] presented a test-driven modeling suite to support test-driven modeling of declarative processes. Mou and Ratiu [8] proposed a method for model-based, test-driven systems development with seemingly limited scope. Dietrich, Dressler, Dulz, German, Djanatliev, Schneider, Yupatova, and Deitsch developed a test-driven agile simulation method and a complex tool chain called VeriTAS (later SimTAny) utilizing UML, SysML, MARTE, UPT, and various tools and frameworks [9][10][11][12][13]. See section 3.3 and 4.4 for further details concerning test-driven modeling and related methods.

RQ 2.2 *Which limitations of existing methods for test-driven systems modeling necessitate modified or new methods?*

RA 2.2 Previous methods focus mainly on unit "testing" and simulating scenarios to demonstrate expected results. Such methods are not likely to capture unintended emergent behavior because only expected results are checked. New methods that monitor selected variables in all states to expose errors that emerge from combining all system parts are therefore needed.

Using simple simulation techniques to verify systems cannot guarantee functionality or performance because erroneous states may not be executed in the scenarios. New methods that utilize formal and statistical model checking to exercise the entire state space are therefore needed.

Previous methods require specialized modeling languages and tool chains that are difficult to learn, use, maintain and evolve. New methods are therefore needed to facilitate test-driven modeling using only existing simple tool that can be applied independently without any need for complicated integration.

A method for test-driven modeling of embedded systems has been proposed in this thesis to overcome the limitations of previous methods. The proposed method is based on formal and statistical model checking and can be used to guarantee functionality and performance for embedded and similar systems. The choice of modeling formalism and tools limits the size and complexity of systems that can be modeled and verified.

A method for test-driven modeling of smart systems has been proposed to overcome the limitations of previous methods (including the abovementioned proposed method for test-driven modeling of embedded systems). The method for smart systems can be used for modeling very large and complex systems. This method utilizes several verification techniques to ensure functionality and performance of basic interactions, to simulate the overall systems with medium to large usage scenarios, and to predict the performance of very large realistic scenarios.

Both proposed methods are limited to systems that can be modeled with the discrete event formalism. Other possible forms of test-driven, model-based systems engineering have been outlined for future work.

RQ 2.3 *What are the advantages, disadvantages and limitations of different verification techniques for test-driven systems modeling?*

RA 2.3 The following techniques can be used for verifying the functionality and performance of systems:

Simulating is used by all reviewed methods to predict system behavior. Unit simulating is used for verifying sub-systems, parts and components. Scenario simulating is used for verifying scenarios of the overall systems. Simulating can scale to very large systems and is mainly limited by simulation time. Simulating does not necessarily exercise the entire state space of the system, so functionality and performance can be estimated but not guaranteed. Simulating is also used in the proposed methods. In the method for TDM of embedded systems, simulating is used for inspection of system states and properties versus time. In the method for TDM of smart systems, simulating is used as the main instrument of obtaining verification data.

Formal model checking has not been used in previous test-driven methods, but it is used in the proposed methods. In the TDM method for embedded systems, FMC is used for verifying the behavior and performance of the overall systems. Verification results can be guaranteed because all reachable states can be analyzed. To avoid state space explosion, much abstracting and simplifying must be done, which limits the fidelity of the models. In the TDM method for smart systems, FMC is used for designing and verifying patterns for guaranteed functionality and performance of basic interactions. FMC cannot be used for verifying the overall system in this case because the size and complexity of the system models would cause state explosion and memory exhaustion during verification.

Statistical model checking has not been used in previous test-driven methods, but it is used in the proposed method for TDM of embedded systems. Here, SMC is used for estimating probabilities and values of systems properties. System functionality and performance can only be guaranteed with a certain (specifiable) probability. SMC was found to be more reliable than simple simulating for finding peak values of properties with large peak-to-average ratios. SMC can theoretically be used for verifying very large and complex systems, but limitations of the UPPAAL-SMC verifier prevent practical and efficient use of this techniques for large and complex smart systems due to slow execution. Other statistical simulators may be faster but have not been tested in this project.

Mathematical forecasting using data fitting and extrapolation techniques allows predicting results for scenarios that are too large and complex to be simulated. The technique requires simulating series of scenarios with fewer users, less complex data, etc., to obtain simulation results for the data fitting and extrapolation. This technique has lower trustworthiness than direct simulating or model checking, but it can be used in cases where no other method can provide answers. Mathematical forecasting is therefore included in the proposed TDM method for smart systems.

Constraint checking can be used for verifying the static structure of system models. Such methods cannot verify the functionality or performance of systems, but they can be used for validating fundamental models. Research concerning such methods is left for future work.

RQ 2.4 *What can be achieved and what are the experiences of using test-driven modeling for design of embedded and smart systems?*

RA 2.4 Different experiences have been reported for previous method. Zhang [2] reported increased productivity and quality when their method was applied on large projects. Zhang and Patel [6] reported threefold productivity increase and less defects, despite several team members lacked prior experience. Zugal et al. [7] reported reduced cognitive load during modeling, increased perceived quality, and improved communication between domain and modeling experts. They also found that the sequential nature of test cases provides intuitive way of modeling. Dietrich et al. [9] applied their method on existing cases and found error that had not been found with previous methods.

The proposed method for TDM of embedded systems was applied to an industrial case, where critical system parameters identified. Knowledge of the system and its parts was also gained during the modeling activities. Easy experimentation and exploration of the design space identified several feasible design variants and excluded others. Further development of hardware and software on doomed concepts was therefore avoided.

The proposed method for TDM of smart systems was applied to an industrial case with 4·X million (confidential) expected acting entities (users, devices, applications, and cloud services). Using the method was straightforward. Significant insight into the critical parameters of the system was gained during modeling. The method allowed easy experimentation to develop a solution that passed verification of all expressed requirements.

RQ 2.5 *What are the advantages of the proposed methods for test-driven systems modeling?*

RA 2.5 The proposed methods use only existing tools that can be used without modifications. There is no need for tool integration, model transformations, or similar activities that otherwise require creation and maintenance of special tools, plugins, meta-models, etc.

The proposed method for TDM of embedded systems utilize formal and statistical model checking. Functionality and performance of system solutions can be guaranteed because the entire state space can be explored when subjected to formal model checking. Latent or hidden errors and unintended emergent behavior can therefore be identified and removed at early development stages. Statistical model checking can show characteristics that are difficult to obtain with other methods, e.g. peak values for properties with large peak-to-average values. The UPPAAL tool that is used with the proposed method includes a strong set of features for debugging models, which is utilized maximally in the proposed method for creating and refactoring models.

The proposed method for TDM of smart systems utilize formal and statistical model checking, simulating and mathematical forecasting. The use of formal or statistical model checking for designing and verifying basic interactions results in solid patterns (with guaranteed functionality and performance) that can be included in the overall system model. Simulating scenarios and monitoring system states and data during simulation, allow precise verification of very large systems with complex interactions. The use of simulation frameworks based on generic programming languages allows modeling of very complex algorithms and data structures, thus increasing the fidelity of the models. Mathematical forecasting based on data fitting and extrapolation enables predicting system performance for scenarios that are too large and complex to be simulated.

RQ 2.6 *What are the disadvantages and limitations of the proposed methods for test-driven systems modeling?*

RA 2.6 The proposed methods is limited to systems that can be described with the discrete event formalism, which excludes truly heterogeneous systems that require continuous time or similar modes of computation.

The proposed method for TDM of embedded systems rely mainly on formal model checking, which limits the method's usefulness. The risk of state explosion and memory exhaustion limits the size and complexity of the models and forces much abstraction and simplification, which reduces the fidelity of the models. Wrong conclusions may arise if the risk of fidelity is ignored.

The proposed method for TDM of smart systems rely mainly on scenario simulating based on models that are created with software development tools. The use of programming languages for modeling caused a tendency to change mindset from systems thinking to solving programming problems. This may hamper creativity and productivity if the risk is ignored.

The use of the SimJava2 simulation framework in the proposed method for TDM of smart systems limits the number of acting entities to approximately 200.000 (depending on the computer and Java version) because the entities are allocated to separate threads. The penalty of switching between threads increases simulation time significantly for large models. An improved simulation framework is required for solving this problem. The implementation of a better simulator is left for future work.

Mathematical forecasting used in the proposed method for TDM of smart systems has low trustworthiness for several reasons. Some forecasts can be shown to be rather imprecise even for scenarios that can simulated. Other forecasts seem more precise but it is not possible to determine the precision for scenarios that are too large and complex to be simulated.

The research has thus uncovered scenarios that have not been sufficiently considered in previous work. Two method have therefore been proposed to fill this gap. A method for TDM of embedded systems has been proposed that utilize formal model checking, statistical model checking and simulations for test-driven modeling of embedded. A method for TDM of smart systems has been proposed that utilize scenario simulating, optional formal model checking, and mathematical forecasting. Potential improvements and future work for the proposed methods have been identified and suggested.

5.2 Evaluating research hypotheses

The research hypotheses (RH) were introduced in section 1.7. The hypothesis evaluations (HE) are described in the following sub-sections based on the research described in chapter 2, 3 and 4.

5.2.1 MBSE tool selection

The following HEs concerning selecting, implementing and using MBSE tools are based on the research described in chapter 2 and they are mostly derived the research answers RA 1.1 to RA 1.6 from section 5.1.1:

RH 1.1 *Several methods for selecting tools that relate to some parts of systems engineering exist but are inadequate for general selection of MBSE tools.*

HE 1.1 Yes, previous methods are tailored towards specific domains. Generic evaluation criteria for the particular domain are typically used. In the general case of selecting systems engineering tools, it is difficult or impossible to define generic evaluation criteria. Existing methods are therefore inadequate for such cases.

RH 1.2 *New methods are needed for systematic analyzing tool requirements to develop relevant selection criteria for MBSE tool selection.*

New methods are needed for evaluating, selecting, and implementing tool solutions to accommodate situations not previously addressed.

HE 1.2 Yes, existing methods uses generic selection criteria for specific domains that cannot be used for selecting various MBSE tools. New methods that include steps for analyzing requirements and for developing tool specifications and evaluations criteria are need for the general case of selecting MBSE tools.

Yes, existing methods can only be used for single-tool scenarios. New methods are needed to accommodate multi-tool scenarios.

Further, existing methods do not include steps to verify or validate the choice of tools. New methods are needed for including such steps.

RH 1.3 *Correlating requirements and tool capabilities to identify non-conformance to different groups of requirements can be used for avoiding tool solutions that cannot be verified.*

Tailoring the elicitation of tool requirements to the individual organization increases the likelihood of obtaining solutions that can be validated.

HE 1.3 Yes, gap analysis was used in the case of selecting a traceability tool. The analysis identified tools that were infeasible (due to lack of essential features) despite the fact that they achieved a high overall rank based on average weighted scores.

Tailoring the requirements elicitation process to the specific situation intuitively increases the likelihood of successful validation. The learning points from the case of selecting a traceability tool selection support this claim. However, the conducted research do not provide further evidence of the claim. More research is therefore needed to evaluate this hypothesis.

RH 1.4 *Using systematic method for selecting and implementing tools increases the likelihood of obtaining good solutions that are accepted by the stakeholders.*

HE 1.4 Experiences gained from the traceability tool selection case indicates that the involvement of stakeholders in the requirements elicitation did lead to the acceptance of the implemented solution. However, the conducted research do not provide further evidence of the claim. More research is therefore needed to evaluate this hypothesis.

RH 1.5 *The research will provide comprehensive methods that allow successful selection, implementation and subsequent use of MBSE tool solutions.*

HE 1.5 Yes, the research has provided a method that include four major steps for justifying and initiating the change project, developing tool specifications and evaluation criteria, investigating tools and selecting best candidates, and implementing, maintaining and evolving the chosen solution.

The proposed can be used for single-tool scenarios, where the best possible candidate is chosen based on ranking. The gap between the requirements and tool capabilities are analyzed to avoid selection of infeasible tools. The case of selecting a traceability tool used this method for successfully selecting and implementing a tool that has been in continual operation since it 2013.

The proposed method can also be used with multi-tool scenarios, where several tools are selected such that they meet all essential requirements. Set theory and simple mathematical operations are used for identifying feasible sets of tools. The method includes classifications of systems and modeling disciplines that are used for obtaining comprehensive sets of requirements for MBSE tools. This selection method was used in the case of selecting modeling tools for identifying a set of tools for systems engineering in the company.

RH 1.6 *The disadvantages of the proposed methods will not cause the organizations to select and implement infeasible tool solutions.*

HE 1.6 The main disadvantage of the method is the amount of work it requires for developing tool specifications and evaluation criteria and for investigating tool candidates. This disadvantage may cause some organizations to use less systematic methods. However, this problem was not experienced in case studies. The conducted research do not provide further evidence of the claim, though. More research is therefore needed for evaluating this hypothesis properly.

5.2.2 Test-driven modeling

The following HEs concerning test-driven modeling are based on the research described in chapter 3 and 4, and they are mostly derived the research answers RA 2.1 to RA 2.6 from section 5.1.2:

RH 2.1 *Several test-driven modeling methods exist but they are inadequate for modeling and verifying hearing systems.*

HE 2.1 Yes, the research identified several attempts at test-driven modeling of software and embedded systems. However, hearing systems have characteristics of both embedded systems and of smart systems. For embedded systems, functionality and performance must be guaranteed, which require formal or semi-formal methods. For smart systems, the tools used for modeling and verifying must be able to handle very large and complex systems. None of the reviewed methods supports all of these needs.

RH 2.2 *New methods for test-driven modeling and verification are needed for allowing system engineers to guarantee system functionality and performance of embedded systems.*

New methods are needed for modeling, simulating and predicting functionality and performance of large and complex smart systems.

HE 2.2 Yes, previous methods focus mainly on unit "testing" and scenario simulating to demonstrate expected results. Functionality and performance cannot be guaranteed without using formal or semi-formal methods that explores all reachable system states. Thus, new methods that include such methods are needed.

For large and complex smart systems, new methods are needed for several reasons. Smart systems are typically too large and complex to be analyzed with formal methods, so verification typically rely on simulation methods, which cannot explore all reachable systems states. New methods that include formal verification of the basic interactions in the system are therefore needed. Prior methods only check the output of the system or its parts, because the methods rely on unit testing or scenario simulating. New methods that also allow asserting internal states are therefore needed. Prior methods typically use specialized modeling languages and tool chains that are difficult to learn, use, maintain and evolve. New methods based on existing modeling languages and tools are therefore needed. All prior methods are limited by simulator capabilities. The size and complexity of smart systems easily challenge the simulators such that the capabilities of the tools are exceed or the simulation time becomes infeasible. New methods are therefore needed to allow full analysis of systems that can be simulated only in reduced or simplified versions.

RH 2.3 *Formal and statistical model checking can be used in methods for test-driven modeling and verification of functionality and performance of medium complex embedded systems.*

Simulating can be used in methods for test-driven modeling and verification of functionality and performance of large complex smart systems.

Some systems are too large and complex to be verified with model checking or simulation. The functionality and performance of such systems can be predicted by mathematical forecasting based on simulation results for smaller and simpler versions of the system of interest.

HE 2.3 Yes, the proposed method for test-driven modeling of embedded system includes both formal and statistical model checking to verify functionality and performance. The methods were successfully used for test-driven modeling and verifying parts of an embedded system for GN Hearing.

The proposed method for test-driven modeling of smart systems uses simulating to verify the overall system model. Assertions are used for evaluating and verifying expressions relating to functionality and performance requirements before, during and after the simulation. The method was successfully applied to a case from GN Hearing, where a smart system was designed and analyzed. Various system parameters were adjusted for best performance and subsequently verified. SimJava2 was used as simulation framework in the case study.

Mathematical forecasting based on data fitting and extrapolation techniques was used in a case study to predict the performance for scenarios that were too large to be simulated. Experience from the case study showed that the precision of such forecasts vary considerably even for scenarios that can by verified by simulation. It is not currently possible to evaluate the precision of the estimates for scenarios that are too large to be simulated. In the case study, some stakeholders found moderately imprecise forecasts acceptable if no other method for obtaining results exists. However, highly imprecise forecasts may lead to wrong decisions. Further research is therefore needed for evaluating the usefulness of mathematical forecasting techniques or for obtaining alternatives.

RH 2.4 *The use of test-driven modeling will increase productivity, improve quality and reduce the cognitive load during modeling.*

The modelers will gain deeper insight into the systems of interest and they will be able to answer questions that otherwise cannot be answered sufficiently.

The modelers will be able to explore the design space and evaluate different versions of the intended systems before starting the actual development of system parts.

HE 2.4 Yes, several researchers found increased productivity, improved quality and reduced cognitive load from using test-driven methods during modeling. The experiences from the case studies also supports these claims. However, productivity, quality or cognitive load were not recorded during the case studies. Furthermore, it was not possible to compare the results of using the proposed methods to the results of using other methods. More research is therefore needed to evaluate this part of the hypothesis properly.

Deep insight into the systems of interest was gained in both case studies. For the embedded system, critical timing parameters were identified and limited into feasible ranges. For the smart system, critical features and interaction schemes were identified and analyzed. Therefore, it was possible to answer engineering questions that could not otherwise have been answered.

Test-driven design space exploration is an integral part of the proposed method for embedded systems. The method was used on a case study to successfully create and verify two alternative solutions to the design problem.

RH 2.5 *The research will produce test-driven methods that has several advantages compared to previous development methods (including previous test-driven methods).*

Combining different verification methods will allow modeling of a wide variety of systems and provide more trustworthy verification results.

HE 2.5 Yes, the research has produced two methods for test-driven modeling. Both methods use only existing tools. There is no need for model transformations or other forms of tool integrations, which eliminate the need for creating and maintaining special tools, plugins, meta-models, etc. The method for test-driven modeling of embedded systems uniquely uses formal and statistical model checking to guarantee functionality and performance.

The method for test-driven modeling of smart systems comprises several techniques for verifying system functionality and performance. Formal model checking of basic interactions produce models that are more trustworthy, without requiring this demanding verification technique for the overall systems. Verifying systems by executing simulations can be used for all types of systems. When the size and complexity of the system of interest exceed the capabilities of simulations tools, mathematical forecasting can be used for predicting systems performance. Combining formal model checking, simulating and mathematical forecasting therefore allow trustworthy verification of various types of systems ranging from small and simple to large and highly complex.

RH 2.6 *The disadvantages of the proposed test-driven methods will not negatively affect creativity, productivity, reliability, or trustworthiness of the verification results. Disadvantages will only concern verification time and required computer resources.*

HE 2.6 The method for test-driven modeling of embedded systems is based on the UPPAAL series of modeling tools. These tools have excellent features for creating and debugging models. Formal and statistical model checking offer highly reliable and trustworthy verification results. The size and complexity of the system models are limited by the capabilities of the tool, though. Much abstraction and simplification must therefore be applied, which limits the fidelity of the models. For larger systems, the method for test-driven development of smart systems is recommended.

The method for test-driven modeling of smart systems is based on simulation of models expressed in a programming language (Java was used in the case studies). A tendency to change the mindset from systems thinking to solving programming problems was observed during the case study. The main disadvantage of the proposed method is the questionable trustworthiness of the mathematical forecasting that are used for predicting performance for systems that cannot be simulated.

Psychological effects such as creativity, productivity, mental loading, etc., were not measured during case studies. More research is needed to evaluate these aspects.

5.3 Observations

The company GN Hearing has successfully developed numerous products for hearing systems for several years. Document-based methods have been used for all types of systems engineering. However, the company has acknowledged a need for model-based methods due to increasing complexity of hearing systems.

Modeling is used in different forms in various departments at GN Hearing for design of antennas, electronics, mechanical parts, electro-acoustical components, etc. Each engineering discipline uses different modeling languages and tools that cannot be integrated easily into system-level models. It is unlikely that this kind of integration will be attempted. MBSE can therefore be introduced without considering restricting dependencies from any modeling legacy.

However, several products for hearing systems are based on platforms that have evolved over time by adding, removing and changing features, components, algorithms and communication protocols. Unforeseen results and effects have occasionally occurred due to the changes. Finding and correcting such problems have sometimes been very difficult and caused much extra unanticipated work. The hope that systems modeling can prevent similar problems in the future has been expressed by several developers from the company.

The potential value of MBSE has been demonstrated to the company by course providers, tool vendors, fellow system engineers in the INCOSE network, and the case studies described in this thesis. Most reactions to these demonstrations were in the following categories:

1. People that perceive the value of MBSE as highly overrated and therefore not worth investing in.

2. People that find MBSE interesting but infeasible due to technological, organizational, cultural or economic barriers.

3. People that are willing to overcome the obstacles of introducing MBSE because they anticipate great value from it.

Some people in the first category argue that documents are much more versatile than models for describing architecture and requirements and for communicating with stakeholders. Prior successes seem to confirm this argument. However, hearing systems are becoming more complex, which necessitates new development methods that allow early identification and correction of potential problems. In this thesis, it is argued that TD-MBSE is a promising candidate for such new development methods.

People in the second category are pointing to concerns that are credible and must be addressed for successful introduction of MBSE. However, the problems seem surmountable and feasible solutions seem possible. The real problem therefore appears to be resistance to change, lack of knowledge, or missing recognition of possibilities. This can be mitigated by providing evidence of the feasibility and by careful planning of the introduction of the MBSE methods and technologies.

People in the third category believe or have witnessed that modeling, simulating and other forms of analyzing and verifying systems can reveal information that otherwise cannot be obtained. Therefore, they are possible supporters that can be utilized for successful introduction of MBSE.

GN Hearing has initiated this PhD project and other activities to include MBSE in future SE work, which indicate that supporters from the third category will promote successful implementation of MBSE in the company. However, a clearly defined strategy is still pending.

5.4 Problems

The method for selecting MBSE tools (section 2.3) has the following disadvantages, problems or limitations:

- The method is very detailed and may lead to decision paralysis or abandonment of the method.
- Knowledge of modeling disciplines is needed.
- Testing of tools may require skills that have not been acquired in advance.
- Comparative evaluation of many tools is time consuming and resource demanding.

The TDM method for embedded systems (section 3.5) has the following disadvantages, problem or limitations:

- Much abstraction and simplification are required to avoid state explosion during verification.
- The UPPAAL modeling language has limited expressiveness, which prevents creating models of high fidelity.
- Verification of mixed-level models results in very long execution traces, which cause memory exhaustion during model checking or simulating.
- The method is limited to systems that can be described in the discrete event formalism.

The TDM method for smart systems (section 4.6) has the following disadvantages, problems or limitations:

- Focus on solving programming issues hampers and interferes with undivided focus on systems thinking, which may result in less elegant system solutions.
- The current implementation of the method uses the SimJava2 framework, which wastes much time on switching between tasks in different threads. This leads to significantly increased simulation times for systems with many entity instances.
- Mathematical forecasting can be very imprecise. The precision may be acceptable at conditions that are close to those that formed the basis for the estimator functions. However, the precision may decrease when the conditions

change considerably from the basis conditions. Extremely high simulation times make it impossible to predict or verify the growing imprecision due to infeasible simulation times.

5.5 Open issues

Three methods have been proposed in this thesis. The method for tool selection exists in two variants. The variant used with single-tool scenarios has been verified in a case study where a traceability tool was selected and implemented in GN Hearing. The variant used with multi-tool scenarios has been demonstrated in a case study for selecting MBSE tools for GN Hearing. However, this variant has not been thoroughly verified, because the final steps of implementing the chosen tool setup in the company have not been completed. Both variants should be subjected to further verification and validation on more case studies in various organizations with different needs. Such additional research should preferably be conducted by independent researchers.

In addition to tool selection, this thesis focuses on TDM methods. Methods have been proposed for TDM of the functionality and performance of embedded and smarts systems. Each TDM methods has been verified in a single case study. Both methods should be verified on more case studies by modeling and verifying different types of systems. The additional research should also be done by independent researcher in this case.

Test-driven approaches can probably be utilized for creating better models in more general modeling contexts than used in this thesis. A few informal experiments showed that constraint checking could be used for verifying SysML or UML models. Combining model templates and test cases expressed in the object constraint language (OCL) can therefore most likely be used as part of a TDM method for FM and EM. However, it is still an open issue how to apply test-driven approaches to all types of model elements in FM, EM and other modeling disciplines. Furthermore, the value of such methods is uncertain and requires more research.

5.6 Future work

More case studies should be conducted for the proposed methods by independent researchers to improve the verification and validation of the methods in a wide range of situations. Setting up and conducting cases to study the proposed TDM methods is relatively easy due to the large availability of cases for which the methods are relevant. However, setting up cases for studying the proposed method for tool selection is more complicated, because it requires significant resources and commitment to complete such cases. Identifying organizations that are willing to conduct such cases to support research may require much convincing. In all cases, much planning and careful execution are required to avoid introducing bias in the results.

GN Hearing has initiated this PhD project to investigate MBSE for potential future use in SE. However, the implementation of MBSE has not yet been commenced even though licenses for a SysML tool have been purchased. There are still issues that must be clarified as part of the future work:

- Is it better to centralize MBSE into the SE department, to use a more decentralized approach, or to use hybrid approaches where modeling and domain experts cooperate across departments?
- Can MBSE be closely integrated with other disciplines in the company to achieve a high level of integrated engineering?
- How should modeling be incorporated into the development processes to improve productivity and quality?
- What should be included and what should deliberately be avoided in the chosen setup to maximize the benefits and to avoid the disadvantages?
- Is it better to hire new employees with modeling skills, to train existing employees, or to embrace both options, if it is decided to implement MBSE?

To implement MBSE after clarification of the above issues, the processes must be defined, the tools must be setup, training (in methodologies, modeling languages and tools) must be provided, projects must be identified, and modeling commenced. During and after completion of projects, the results should be evaluated for potentially improving processes, methods, modeling languages and tools for future use.

An MBSE strategy that include TDM has not been defined yet. The SE department is mainly working with requirements managements, architecture descriptions, and systems testing. *System design*, which is the activity that most significantly can benefit from TDM, is rarely performed by the SE department but by specialty engineering disciplines, e.g. software developers. To benefit from TDM, the company must therefore either:

- Centralize system design into the SE department, or
- Distribute TDM into the specialty engineering departments.

It is also possible to embrace both approaches, where expertise is first build in SE and later distributed to specialty engineering disciplines.

Diagrams expressed in notations similar to SysML or UML are currently used in the company for developing software sub-systems. Simple drawings tools have been used for creating such diagrams. Without using proper modeling tools, there are no means of checking for correct notation. Examples of diagrams with incorrect notations have been observed repeatedly, which increases the risk of wrong interpretation and thus wrong implementations. Proper modeling tools and training should therefore be provided for software architects and developers. It should be investigated whether the software departments can use the same tools as SE or if they might benefit from using different tools. Using the tool selection method presented in chapter 2 may help identifying feasible solutions. To increase the quality of modeling within software departments and other specialty engineering disciplines further, it should be investigated if TDM can be adapted and used for improved productivity and quality of sub-systems.

Creating or improving simulators, verifiers, model editors, and other tools for MBSE and TDM should be part of the future work:

- Front-end for creating UPPAAL queries from requirements expressed in less rigid (i.e "human") language. Alternatively, a tool that translates queries into human language may help engineers to ensure that queries express the intended requirements.

- Model transformation tools to convert between SysML models and UPPAAL models for templates and global declarations. Using SysML as the primary modeling language has several benefits such as allowing traceability to requirements of originating stakeholders, providing reporting features, providing a uniform interface for all MBSE activities, etc.

- Model transformation tools to convert between SysML models and SimJava2 models such that SysML can be used for providing an overview of interactions and message exchanges between entities, which can otherwise be difficult to maintain.

- Improving and adding new features to the UPPAAL tools. Pre-processor or compiler directives for separating formal and statistical model elements may ease modeling and verification. Simplified clock modeling may also have usability benefits. Break-points for the interactive simulator will allow easier debugging of scenarios with long and complex execution traces. Better memory management is needed for allowing simulation and verification of larger and more complex systems.

- Substituting SimJava2 with a more efficient and faster simulation framework should be undertaken to allow simulation of scenarios with more users, devices and other entities than currently possible. It should be investigated if existing frameworks (e.g. the core of CloudSim) can be adapted and used with the proposed TDM method for smart systems. The feasibility of distributed computing, use of super computers, and other techniques to reduce simulation time significantly should also be investigated and used if feasible.

The proposed TDM methods can only be used with systems that can be expressed in the discrete event formalism. However, many embedded and smart systems contain elements that can only be expressed in other formalisms such as continuous time. To achieve truly holistic test-driven design of such systems, new methods are needed. The future work should therefore investigate and propose methods for TDM of heterogeneous systems, where parts can be expressed in various formalisms, and where the overall system can be verified by co-simulating models that require different modes of computation.

The future work should finally investigate test-driven methods for other MBSE disciplines. Using model templates and constraint checking with existing modeling tools may be feasible. The feasibility of such and other TD-MBSE methods should be thoroughly investigated before they are adopted by industry.

5.7 Impact

The PhD project has provided the researcher (me) with a unique opportunity to learn and explore the fields of test-driven and model-based methods in the systems engineering domain. The new skills that I have required during the project have enabled me to undertake assignments and job functions that I would otherwise not have been qualified for.

GN Hearing has gained insight into the various MBSE disciplines. The company has been provided with methods for selecting feasible modeling setups that support their actual needs. TDM methods for design of embedded and smart systems have also been provided. Using the acquired knowledge and methods has the potential to foster creativity, improve productivity, and reduce risks and costs of developing more complex and competitive system solutions. Substituting or augmenting prior document-based development methods with test-driven modeling may enhance the company's reputation for professionalism.

Modeling, programming, simulating, verifying, and similar engineering activities are probably more attractive to engineers and similar personalities than writing documents. Existing and future employees at GN Hearing are therefore likely to perceive the content of their jobs with greater satisfaction. The acquisition of new skills and competencies in a field of increasing importance makes employees more attractive and improves their career prospects.

The embedded systems engineering (ESE) section at the Department of Applied Mathematics and Computer Science (DTU-Compute) at the Technical University of Denmark (DTU) has gained increased insight into the various MBSE disciplines and TDM. The acquired knowledge and identification of future work can be leveraged for launching new research, master project, and PhD positions.

The project has provided a thorough and comprehensive method for selecting MBSE tools for the general industry. Using this method is likely to increase the success rate of selecting the modeling tools and technologies that fit the actual needs of the companies. Adopting of MBSE is therefore more likely to be successful.

The proposed methods for TDM of embedded and smart systems can be applied by many industries and a large range of companies. The methods may even be adapted to different domains than embedded and smart systems. All types of systems where subsystems, parts, components and individuals exchange messages or objects as discrete events may be designed by using the proposed TDM methods. The companies and their employees are expected to gain the same type of benefits as described above for GN Hearing above.

Several tool providers are marketing modeling technologies that facilitate MBSE to a broad range of users. Such modeling technologies are used either for low-level modeling and simulation (e.g. Simulink) or for high-level system design with a focus on communication with stakeholders (e.g. SysML tools). Test-driven methods are used in neither domains. Adopting the proposed or similar test-driven methods may increase the value of both low- and high-level modeling. Tool providers therefore have the possibility of adding features that make their products more attractive for potential customers.

The methods proposed in this thesis thus affect many different types of companies, organizations and branches of society. The proposed methods therefore promote the technological trends and enables the development of large complicated systems for the benefit of all citizens.

5.8 Conclusion

This thesis has addressed two major questions:

RQ 1 How can organizations select and implement tools for (model-based) systems engineering?

RQ 2 How and to what effect can test-driven methods be applied to model-based systems engineering?

RQ 1 and its 6 sub-questions were addressed in the research that are presented in chapter 2. RQ 2 and its 6 sub-questions were addressed in the research that are presented in chapter 3 and 4. Detailed answers to the sub-questions were presented in section 5.1.

For both RQ 1 and RQ 2, the research uncovered scenarios that had not been sufficiently considered in prior work. Novel methods were therefore proposed to fill the gaps.

For RQ 1, a method was proposed for selecting, implementing and using MBSE tools. The proposed method include 4 major steps to justify and initiate the change project, develop specification and evaluation criteria, investigate tools and select best candidates, and implement, maintain and evolve the solution. In tool selection situations, there are generally two cases: (1) Select the best tool and accept its shortcomings (single-tool scenario). (2) Select a set of complementary tools that fulfills all essential requirements (multi-tool scenario). Prior work only considers the single-tool scenario. The proposed method accommodates both single-tool and multi-tool scenarios. For single-tool solutions, it uses ordinary ranking to identify the best candidate. However, to avoid selecting infeasible tools, the method includes a novel step of analyzing the gaps between requirements and tool capabilities. For multi-tool solutions, it uses set theory and simple mathematical operations to identify feasible sets of tools that meets all essential requirements. In cases where not all requirements can be met, subsequent ranking and analyzing the gaps can be used for identifying the best set of tools. Both variants of the method were tested on separate case studies with good results.

For RQ 2, a method for test-driven modeling of embedded systems was proposed. The method was developed from traditional TDD. TDD steps were transformed and new unique steps were added. The method uses formal model checking for verifying the system behavior. Statistical model checking is used for estimating probabilities and values of system properties and quality attributes. Simulations are used for inspecting system states and properties. The method includes steps for design space exploration such that variants of basic design solutions can analyzed and compared. The method was tested on a case study with good results.

For RQ 2, a method for test-driven modeling of large and complex smart systems was also proposed. The method uses formal model checking for designing and verifying the basic interactions in the system under analysis. However, the complete system

Table 5.1: Evaluations of research hypotheses.

Hypothesis:	1.1	1.2	1.3	1.4	1.5	1.6
Evaluation:	Yes	Yes	Yes	Likely	Yes	Maybe
Hypothesis:	2.1	2.2	2.3	2.4	2.5	2.6
Evaluation:	Yes	Yes	Mostly	Maybe	Yes	Maybe/No

is typically too large and complex for formal model checking. Simulation techniques are therefore used for designing and verifying the overall system. Statistical model checking was discarded because current simulators are too slow for realistic use. The method include mathematical forecasting for predicting performance of systems are too large or complex to be simulated. Smaller or simpler versions of the systems are simulated in series with varying parameter values to obtain results that can be subjected to data fitting and extrapolation. The mathematical forecasting is occasionally imprecise, which is undesirable but acceptable when no other method can be used for predicting functionality or performance. The method was tested on a case study with good results.

Limitations, observations, learning points, advantages, disadvantages, and potential improvements were identified and future work was suggested for all proposed methods, see section 2.8, 3.7 and 4.9 for details. Observations, problems, open issues, and future word were summarized and further discussed in section 5.3, 5.4, 5.5 and 5.6, respectively.

The research hypotheses were evaluated in section 5.2. An overview of the evaluations is shown in table 5.1. RH 1.4, RH 1.6, RH 2.3, and RH 2.4 could neither be fully confirmed nor fully rejected without more research. RH 2.6 was partially rejected for the method for test-driven modeling of smart systems, because the observed tendency to shift focus from systems thinking to solving programming issues had a negative effect on the creativity and productivity. More research is needed to determine the seriousness of this problem. The method for test-driven modeling of embedded systems does not suffer this problem. Seven out of twelve hypotheses have been fully confirmed. It is not unlikely that the remaining hypotheses (excluding RH 2.6) can be confirmed by further research. However, the purpose and scope of the project did not allow conducting this additional research, which is therefore allocated to future work.

The project has provided methods and guidelines that allow GN Hearing to commence the transition from documents to models for their systems engineering activities. The proposed method for selecting, implementing and using MBSE tools can be leveraged by GN Hearing and similar companies to analyze their MBSE needs and to select feasible tool solutions. The propose method for test-driven modeling of embedded systems can be used by GN Hearing and similar companies for designing and verifying systems of medium size and complexity. The propose method for test-driven modeling of smart systems can be used by GN Hearing and similar companies for designing and verifying very large and complex systems.

It is the author's hope that GN Hearing and similar companies will embrace the proposed methods and guidelines to commence the transition in systems engineering from documents to models.

5.9 References

[1] Allan Munck
"Model-Based Systems Engineering Guidelines."
Technical report, Technical University of Denmark, Department of Applied Mathematics and Computer Science.

[2] Yuefeng Zhang.
"Test-driven modeling for model-driven development."
Software, IEEE 21.5 (2004): 80-86.

[3] Susumu Hayashi et al:
"Test driven development of UML models with smart modeling system." In: *"«UML» 2004 - The Unified Modeling Language. Modeling Languages and Applications."*
Springer Berlin Heidelberg, 2004. 395-409.

[4] Esteban Robles Luna, Julián Grigera and Gustavo Rossi.
"Bridging test and model-driven approaches in web engineering."
International Conference on Web Engineering. Springer Berlin Heidelberg, 2009.

[5] E. R. Luna, J. I. Panach, J. Grigera, G. Rossi and O. Pastor.
"Incorporating Usability Requirements In a Test/Model-Driven Web Engineering Approach."
J. Web Eng. 9.2 (2010): 132-156.

[6] Yuefeng Zhang and Shailesh Patel.
"Agile model-driven development in practice."
IEEE software 28.2 (2011): 84.

[7] Stefan Zugal, Jakob Pinggera and Barbara Weber.
"Creating declarative process models using test driven modeling suite."
Forum at the Conference on Advanced Information Systems Engineering (CAiSE). Springer Berlin Heidelberg, 2011.

[8] Dongyue Mou and Daniel Ratiu.
"Binding requirements and component architecture by using model-based test-driven development."
Twin Peaks of Requirements and Architecture (Twin Peaks), 2012 IEEE First International Workshop on the. IEEE, 2012.

[9] Isabel Dietrich, F. Dressler, W. Dulz and R. German.
"Validating UML simulation models with model-level unit tests."
Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (p. 66). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[10] Anatoli Djanatliev, W. Dulz, R. Germana and V. Schneider.
"Veritas - A versatile modeling environment for test-driven agile simulation."
Proceedings of the 2011 Winter Simulation Conference (WSC), (pp. 3657-3666). IEEE, 2011.

[11] Vitali Schneider and Reinhard German.
"Integration of test-driven agile simulation approach in service-oriented tool environment."
Proceedings of the 46th Annual Simulation Symposium. Society for Computer Simulation International, 2013.

[12] Vitali Schneider, A. Yupatova, W. Dulz and R. German.
"How to avoid model interferences for test-driven agile simulation based on standardized UML profiles (work in progress)."
Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative (p. 35). Society for Computer Simulation International, 2014.

[13] Vitali Schneider, A. Deitsch, W. Dulz and R. German.
"Combined Simulation and Testing Based on Standard UML Models."
Principles of Performance and Reliability Modeling and Evaluation. Springer International Publishing, 2016. 499-523.

[14] Orlena Gotel and Patrick Mäder.
"How to select a requirements management tool: Initial steps."
2009 17th IEEE International Requirements Engineering Conference. IEEE, 2009.

[15] Orlena Gotel and Patrick Mäder.
"Acquiring tool support for traceability."
Software and Systems Traceability. Springer London, 2012. 43-68.