UNIVERSITY OF TROMSØ UIT

Faculty of Science and Technology
Department of Computer Science

Model-Driven Software Development for Continuity of Care Information Systems

Toolchain design and evaluation

Ståle Walderhaug

A dissertation for the degree of
Philosophiae Doctor

November 2012

Abstract

As more people receive care services in their homes, the importance of information systems supporting continuity of care increases. To develop such information systems, major interoperability issues must be addressed with agreement on domain standards and technical platforms. Lately, service oriented architectures have become a popular technical solution for sharing information and services between systems in healthcare. However, the reuse of domain software service is limited and the standardization processes have just started.

Using models for designing software is best practice in software engineering, but few developers are utilizing the models for code generation with model-driven development tools. The key benefits claimed for model-driven development clearly fits the challenges of developing reusable software service in healthcare. Nevertheless, the scientific knowledge on how to utilize model-driven development for creating standardized and reusable software in healthcare is scarce. Which features of model-driven development are useful? Should tools be adapted to the healthcare domain? Which type of software services can be reused? These are questions being addressed in this thesis.

This thesis summarizes six years of design, development and evaluation of a model-driven development toolchain and design of software services for continuity of care. The overall research method has been design science, with a strong focus on creating and evaluating the core artefact: the ModelHealth Toolchain. Three iterations with toolchain design and assessments were deemed necessary to be able to draw valid conclusions about tool design and development mechanisms.

A significant part of the work was carried out within the European research project MPOWER. This allowed for both toolchain evaluations with professional developers, and reference implementation of the identified software services. The services were used as a foundation for development of two pilot systems that were evaluated with end users.

Based on the contributions from ten papers, a total of eight findings connected to the three research questions have been identified. To summarize, the research has shown that model-driven development can aid developers in creating healthcare software services, given that the modelling tool fulfills some important requirements. The tool should be easy to use, provide project structure and process assistance, and support core traceability services such as navigation and coverage analysis.

The ModelHealth Toolchain also successfully demonstrated incorporation of domain knowledge such as continuity of care concepts from an international standard through UML Profiles. The presentation of this domain knowledge must be carefully designed so that it allows for easy and correct use by the developers.

The overall contributions of this thesis are

- A set of reusable software service designs for continuity of care, provided as open source

- The ModelHealth toolchain supporting model-driven development of continuity of care web services, including reusable model elements and UML Profiles
- A set of recommendations on how to tailor a model-driven development toolchain for domain such as continuity of care.

A final important contribution is the comprehensive documentation of a complete design science research project, where all the three research cycles are involved. This increases the body of literature on design science research in software engineering.

Preface

My first project at SINTEF was in the field of healthcare informatics. In the EU-project TelemediCare (1999-2002) we developed a novel system for home monitoring of children with chronic diseases. We spent significant resources on developing cross-platform software and integrating with wireless sensors and dial-up networking. In the end we managed to reach a state where the system could be evaluated with the SABH unit at the Karolinska hospital in Stockholm, Sweden. We concluded that even though the system was well received by the stakeholders involved, we needed a major refinement.

In 2002-2006 I was the project manager for a collaboration project between the Norwegian Military's Joint Medical Services and the US Army Telemedicine and Advanced Technology Research Center (TATRC). In the project we developed a PDA and tablet based system for patient tracking and medical treatment documentation and sharing. The goal was to replace the paper-based solution where the documentation rarely made it to the patient's medical record, and to facilitate the maintenance of the soliders' "complete longitudinal" medical record.

In the project, I worked closely with the people at the medical battalion in Norway as well as TATRC. They were quite enthusiastic about our solution for information flow in the "Medical Evacuation Chain", and in December 2002 we evaluated the system at the military exercise "Cooperation" in northern Norway. The conditions could not have been better – or worse. It was dark, rainy, icy, windy and noisy. The evaluation was a success, so the system could be refined. Of course the system was not perfect the first time, but the rather thorough evaluation gave us invaluable feedback. I guess this is where I really learned that software system evaluation is way more complicated and important than the impression you get when you learn about the classic waterfall model at the university.

An important part of the work with the joint medical services was to represent Norway in the NATO Telemedicine Expert Panel. During these meetings I presented the results from our systems development and I learned that using UML diagrams was a powerful tool for discussing concepts and solutions across disciplines – military, medical and technical.

John Ivar Brevik, MD, was the head of military medical research in the Norwegian Joint Medical Services, and a scholar. John Ivar and his colleague Major/MD Terje Sagen had a principle about "one soldier – one medical record", and together they had led the implementation and deployment of the Norwegian Military EHR (called SANDOK) that was unique in NATO at the time. It was also during long discussions with John Ivar that I decided (at a restaurant in Tampa, Florida, April 2004) to start my doctoral education. The research topics we identified were continuity of care and domain specific model driven software development. My boss at SINTEF gave me permission to move to Tromsø to cooperate with the medical battalion, and connect with the telemedicine people at the University of Tromsø and Norwegian Centre for Telemedicine.

Unfortunately, in 2005 the research funding for John Ivar's office were reduced to zero. Ironically, the reason for killing the research funding was a (yet another) software integration project that failed.

Almost one year later (October 2006), after working intensively with model-driven traceability in the ModelWare EU-project, SINTEF started the MPOWER project where I became the technical manager and work package leader for architecture and development approach. The main objective of the project was to create model-driven development approach and a SOA-based middleware platform for homecare. My boss and I agreed that I could revitalize my doctoral studies, and in March 2007 I was enrolled as a PhD student at computer science department at the university of Tromsø.

My roles as technical manager and leader of architecture and development approach were quite challenging but also gave me the opportunity to test concepts and implement solutions that normally would have been too resource demanding for a doctoral project. Being responsible for the system architecture and designs, I was involved in intense and fruitful discussions, with highly skilled computer scientists and programmers in Croatia, Austria, Cyprus and Spain. It was really motivating to work with the people in the MPOWER project.

I must also include that having Marius Mikalsen, a very good friend, colleague and researcher as the project manager of MPOWER, made it possible and inspired me to keep the focus on research during the project's lifetime. Marius and I share the same view on systems development and evaluation, and we've had numerous discussions on evaluation methods and design science.

The evaluation of the SOA-based care systems in Norway and Poland were led by Torhild Holthe at the Norwegian Center for Aging and Health, and Dariusz Duplaga, MD at the Jagiellonian University Medical College in Krakow, Poland. In the 1-year trial period for the Norwegian system, I had almost daily conversations with Torhild about deployment, technical challenges and user evaluations. I also accompanied a visiting nurse at a visit to the most active participant (a 82 year old lady) in Trondheim, Norway. When I saw how positive influence our "simple" system was on the activities of daily living for the old lady, I could for the first time in my work as researcher clearly see the relevance of my research efforts. Almost a year later I was in Lubomierz, Poland spending one week installing and testing the sensor-centric SOA system in a nursing home for elderly. This was another strong experience for a computer science researcher, where it once again became clear to me that participations in evaluations in realistic environments should be a mandatory activity for all healthcare software system developers.

The design and development of the reusable software services and the development toolchain is well documented in the papers included in the thesis. Moreover, it should be emphasized that a key factor contributing to the toolchain development was the access to students at the computer science department at the university in Tromsø. In parallel with the MPOWER project, I got the opportunity to carry out evaluations with master students as part of their medical informatics and software engineering courses. These evaluations were of utmost importance for the evolution of the ModelHealth toolchain.

In summary, I will say that the projects I've worked in, my roles in these projects, the project partners, specific episodes in the project work, and people I've had the pleasure to work with, have given me knowledge that I want to develop and share with other students, researchers and stakeholders in the domain. This thesis presents a complete design and evaluation process of a model-driven development toolchain for the care domain. The toolchain is applied to web services design of reusable domain software services that is further utilized by two pilot systems deployed to Norway and Poland.

I hope that the research results, the development approach and the approach to software service reuse will inspire you to join me in the work towards a more efficient way to implement continuity of care.

Acknowledgements

This thesis is the result of not only reading papers, writing papers, project teamwork, but also several factors outside my professional arena.

During this doctoral study period, I've gotten two fantastic and energetic boys with the woman in my life – Ann-Katrine. I have no problems admitting that sleepless nights, numerous ideas and unlimited creativity from my two boys, Adrian and Benjamin, have challenged the work process. However, the same factors have given me more energy and most of all, perspective on life. Ann-Katrine, you have put up with a lot from my side in this period, but I've had your unlimited support at all times, and many times you've assisted me in regaining focus on the correct objectives. Without you I don't think I could have done this.

Neither the work nor the thesis would have been completed if it were not for my good friend and colleague Dr Erlend Stav. Erlend, your combination of excellent architecture and design skills, programming skills and thoroughness, together with, to my knowledge, unlimited patience, must be unique. When I contact you for advice or guidance, I will get a highly qualified and nuanced answer, before I expect it. This is, and has been of utmost importance.

During the PhD project, we have established a highly competent research group for healthcare informatics at SINTEF ICT. Led by Marius Mikalsen, the group with Dr Erlend Stav, Dr Babak Farshchian, Dr Anders Kofod-Petersen and myself has developed a significant portfolio of research projects in healthcare informatics domain. We've had important discussions and I've gotten strict review on my work. It is a pleasure to work with you. I also strongly appreciate the flexibility and trust from my research director at SINTEF, Eldfrid Øfsti Øvstedal. In a rather complicated project and funding situation you have organized project staffing and resources so that I could complete my studies.

The work on the ModelHealth toolchain could not have been done without the support from the University of Tromsø. I wish to thank Professor Gunnar Hartvigsen for supervision and facilitating my work with students at the university. I will also express my appreciation to Dr Johan Gustav Bellika at the Computer Science Department for interesting discussions and cooperation on student project assignments. Being a part of the eHealth PhD student environment in Tromsø has inspired me to work hard.

I guess I've always been focused and worked hard to reach my goals. A great acknowledgement goes to my parents who have given me all opportunities to develop my skills and follow my desires. You have supported me during sports and in all phases in my life and I'm forever grateful for the foundation you have provided. I've always known that you are proud of me, regardless of my achievements.

I'm grateful for having so many nice people around me.

Abbreviations

AAL	**Ambient Assistive Technology**
CIM	**Computation Independent Model**
FDA	**Food and Drug Administration**
HL7	**Health Level 7**
ICT	**Information and Communication Technology**
MDA	**Model Driven Architecture**
MDD	**Model-Driven Development**
MDE	**Model-Driven Engineering**
PIM	**Platform Independent Model**
PSM	**Platform Specific Model**
SOA	**Service Oriented Architecture**
SOAP	**Simple Object Access Protocol**
WSDL	**Web Service Description Language**
XML	**eXtensible Markup Language**

Content

List of figures

List of tables

Part I: Summary of the PhD dissertation

Chapter 1 Introduction

1.1 Background for the research

This chapter will introduce the challenges addressed from a care and a software engineering perspective. A mode detailed presentation is provided in chapter Chapter 2.

1.1.1 An ageing population requires new care models

The European countries are facing a great challenge in dealing with a steadily aging population. The 2009 Aging Report projects for 2060 that *"Population structures become increasingly dominated by old people rather than young"* (European Commision 2009). In the period until 2060, it is projected that the working population (age 15-64) will drop by 15 per cent within the EU (starting from 2010), and the number of elderly aged 65 or more will double. The total population will only have a slight increase. Despite a trend of increasing employment rate of women, there will be a shortage of labor to provide care for the elderly.

To maintain the same care service level as of today, there is a need for new care concepts. Assistive services and new innovative information and communication technologies are gradually becoming commercially available, opening for new care concepts that may support elderly and people with cognitive impairments and dementia in living independently at home. To be able to provide optimum care and management of the users, timely access to updated and complete information is essential. Herein lies the problem of providing "continuity of care".

1.1.2 Continuity of care and supporting independent living

The EU's IST programme glossary defines continuity of care to be:

"*The co-ordination of care received by a patient over time and across multiple health-care providers*."[1]

Haggerty et al provide a more extensive definition. They define three types of continuity of care: informational, management and relational (Haggerty et al. 2003). The two first are the most relevant for the scope of this thesis, and are defined as:

[1] IST Glossary, available online at:
http://www.cordis.lu/ist/ka1/administrations/publications/glossary.htm

- *Informational continuity*—The use of information on past events and personal circumstances to make current care appropriate for each individual
- *Management continuity*—A consistent and coherent approach to the management of a health condition that is responsive to a patient's changing needs

Informational and management continuity are major concerns for systems supporting care coordination and independent living. New technology can empower people with age related disabilities to stay active and avoid institutionalization, hence improving quality of life for the elderly and reducing resource demands from health and social care services. Independent living has been investigated in the BRAID (Bridging Research in Ageing and ICT Development) project saying, "*independent living is characterised as being dependent on a safe, secure and suitable environment. A wide range of assistive technologies has been identifiedin this area including: assistive home-based technologies, living status monitoring, agenda manager, mobility aids (including driving), companion robots and well-designed human-machine interfaces that facilitate the use of technologies in general." (BRAID Project 2012)*

From a technological perspective, developing assistive services and information systems that support independent living should focus on standardization and interoperability - compulsory requirements but also a challenge for the developers. System developers must implement a set of agreed standards so that the system can exchange information and reuse services from other systems. Interoperability remains one of the biggest challenges in healthcare information systems development. (European Commision 2008; Grimson et al. 2000; Brailer 2005). Coping with this challenge is a costly process, but the potential savings and improved quality are high. Walker estimated that in US alone, a fully interoperable health information systems would save nearly 80BN$ (Walker et al. 2005).

An important activity on dealing with interoperability is the development of information and communication standards for health-related information. Standards development organizations (SDOs) such as CEN TC251, CONTINUA Alliance, HL7, ISO, OMG and OpenEHR work hard on providing standards that can assist in the development of interoperable systems. With only a few exceptions, these standards are provided to the developers as documents that must be read and interpreted as a part of the development process. The Integrating Healthcare Enterprise (IHE) provide "interpretation documents" through their "IHE Profiles" that assist developers in making the correct decisions. To further verify "correct" interpretations of the standard specification documents, events such as the IHE Connectathon[2] and the Continua Plugfest [3] are organized annually. Here developers can test their implementation against other systems and solve integration issues. Other solutions that verify correct interpretations of standards include XML message checking through online test services. XML based documents such as those based on HL7 CDA

[2] IHE Connectathon homepage: http://www.ihe.net/Connectathon/index.cfm

[3] Continua Alliance plugfest homepage: http://plugfest.continuaalliance.org/

can become complex, and extensive testing is required to ensure full interoperability between the systems.

1.1.3 Interoperability, reuse and service oriented architectures

Developing large software systems such as a healthcare information system is a complex process. Especially when the system needs to integrate with an existing and often multiplatform, multi-standard and proprietary infrastructure. Adding a diverse set of stakeholders to this picture, as is the case for healthcare, makes the development process even more exposed for delays and overspendings.

In 1968 (Naur,Randell 1968), software engineers started talking about "software crisis". Since then, several reports have been published on how software projects run over time, run over budget, do not meet the requirements and even fails to deliver at all, e.g., the Standish CHAOS report (Standish Group International 1994). Boehm summarizes the history of software engineering and presents a view of 20th and 21st century software engineering in (Boehm 2006) and illustrates the core trends in a block diagram along a timeline. From the SAGE methods for hardware engineering in the 1950's through the "code-and-fix" period in the 1960's, the waterfall process is the main focus in the 1970's. The 1980s are dominated by 4GL and object-oriented methods, whereas domain-specific architectures and enterprise architectures are introduced 1990's. Finally, Boehm ends up with service-oriented architectures, agile methods and model-driven development in 2000-2010. He foresees global connectivity and massive systems of systems for 2010's. The trends are influenced and formed from the need to support evolvability, reusability, scalability, integration and rapid change.

Reusability and scalability of software components and services across systems and organizations has received much attention since the specification of the service-oriented architecture (SOA) reference model (OASIS Open 2006). OASIS describes SOA as "*a paradigm for organizing and utilizing distributed **capabilities** that may be under the control of different ownership domains*." Furthermore, they claim the "*value of SOA is that it provides a simple scalable paradigm for organizing large networks of systems that require interoperability to realize the value inherent in the individual components*." This value proposition fits the needs of healthcare information systems perfectly, and many national and international strategic plans have adopted SOA as the integration platform (Kawamoto,Lobach 2007; Nasjonal IKT 2011) or as the technological foundation (European Commision 2011).

The main focus for relevant healthcare SDOs are not on reusable service specifications, rather on information models, coding and message design. However, initiatives such as the Health Level 7/Object Management Group (HL7/OMG) Healthcare Service Specification Project (HSSP) project (HSSP Project 2007) seek to find reusable services and implement them through a defined process. Their current service projects include entity management, resource identification, common terminology, decision support, and directory services for providers. These services represent core services of a healthcare network backbone, but are not sufficient as building blocks for full-scale information system.

The concept of providing a reusable set of "middleware" software service specifications in healthcare is supported by ISO TC215 in the 12976 Health Information Services Architecture (HISA) standard (ISO/TC215 2009). HISA is divided into enterprise, information and computation viewpoints and provides an abstract framework that assists developers in the design of information models and service interfaces. The framework is at a high level and has not received much attention from the domain system developer. Even though the standard specifies naming conventions and information model concepts, there is to my knowledge no tool support for applying it.

Rine and Nada did an empirical survey on software reuse where they found the leading indicators of successful reuse capability to be *"product-line approach, architecture which standardizes interfaces and data formats, common software architecture across the product-line, design for manufacturing approach, domain engineering, management which understands reuse issues, software reuse advocate(s) in senior management, state-of-the-art tools and methods, precedence of reusing high level software artifacts such as requirements and design versus just code reuse, and trace end-user requirements to the components that support them."* Their studies also found that reuse of software decreased the level of development effort and time, increased product quality and shortened time to market. (Rine,Nada 2000)

Investigations on reuse and SOA reveals that there are still many challenges in achieving service reuse, primarily because of poor documentation of functionality, quality and underlying requirements (Dan et al. 2008). So, despite the specification of a SOA reference models and applicable protocols, the challenges of reusing software components persist. In 1992, Krueger did a survey on software reuse trying to find out why software reuse is difficult. One of his key findings, also addressing the challenges described by Dan et al., is what he calls "*cognitive distance*" (Krueger 1992). This is the effort required to reuse software in a new development phase or in another project. Krueger states that the ideal solution to reduce the cognitive distance would be a technology that allows the software developer to:

"quickly be able to select, specialize, and integrate abstraction specifications that satisfied a particular set of informal requirements, and the abstraction specifications would be automatically translated into an executable system." [ibid.]

The "technology" Krueger predicts could take the form of an integrated development environment with domain support for reusing abstract specifications of domain concepts.

1.1.4 Model-driven development tools and healthcare

In 1987, Fred Brooks wrote his paper "No Silver Bullet" where he stated: "*I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems. If this is true, building software will always be hard. There is inherently no silver bullet."(Brooks 1987)*

"Abstract specifications" of underlying concepts, and the ability to create complete "conceptual constructs" is maybe the main value of using modelling languages for

software engineering. Model-driven development (MDD), has been around for many years, though in the context of computer assisted software engineering (CASE) tools in the early nineteen eighties (Iivari, 1996; Lundell and Lings, 2004; Sharma and Rai, 2000). Today, the majority of software development teams use some kind of modelling language to design the software artefacts to be developed (Hutchinson et al. 2011). Most of these languages are related to the Unified Modeling Language (UML), either as specializations or languages built on the same principles (ibid.). Unfortunately, UML is not used in the way it was originally planned (Fowler 2003; Hutchinson et al. 2011) – namely as a modelling language allowing software developers to create abstract designs that can be used as input to the production of executable code.

To facilitate more comprehensive use of UML to include model-driven development, the Object Management Group (OMG) has defined the Model-Driven Architecture (MDA) approach. However MDA or MDD in general, has not yet created the paradigm shift in software engineering that many people hoped for. Iivari explains the "failure" of CASE/MDD tools by looking at the high tool expectations and the missing return of investment (Iivari 1996).

Unfortunately, MDD tools are still a part of the reason why MDD adoption is still low. The immaturity of current MDD tools has been identified as a problem in several empirical studies (Iivari 1996; MacDonald et al. 2005; Mohagheghi,Dehlen 2008; Staron 2006). To deal the MDD tool problems, it is possible to combine several tool components into a **model-driven toolchain** that that ultimately can provide required functionality. Building a proper toolchain is a challenging task and Mohagheghi et al conclude that there *"is no tool chain at the moment and companies must integrate several tools and perform adaptation themselves" (Mohagheghi et al. 2009b).*

The promises of the MDA/MDD technology are highly desirable for most application domains: *"The three primary goals of MDA are **portability**, **interoperability and reusability** through architectural separation of concerns"* (Miller and Mukerji, 2003). For MDD to be successful, it is eminent that the quality of the models being developed is satisfactory in terms of the goal of modelling. Whereas UML is the most used language for modelling in a MDD environment, other languages are supported and UML can be extended with the use of UML Profiles. In (Krogstie 2012), Krogstie argues for modelling languages adaption to increase the language appropriateness:

"In many cases, the modelling language chosen is not appropriate for representing the knowledge on the domain, thus making it very difficult to achieve completeness. One important activity to address this is the adaptation of the meta-model of the modelling language used to suit the domain, both by adding concepts, but also by removing concepts (temporarily) from the language if they are not relevant for the modelling of the particular domain."(page 231)

MDD delivers many useful features to SOA development, e.g. focus on message design, interface design and domain modeling with use cases, sequence and activity diagrams. In 2005, Johnson published an article where he describes how SOA concepts can be incorporated into UML using UML profiles (Johnston 2005). The profile includes UML representations of important concepts from the SOA reference model (OASIS Open 2006), making them available to software designers using a UML modelling tool. In 2009, the SOA and UML work was standardized through

OMG's SoaML (Object Management Group 2009). The standard is supported by some MDD tools, but with various degree of completeness.

1.1.5 Research gaps

The introduction points to challenging areas where substantial research efforts are spent. Still, there seems to be gaps in the research that should be investigated.

From the care perspective, the demographic trends with an aging population and shortage of labor will require new care models and increased focus on continuity of care and independent living. There are several barriers to providing continuity of care, in which the lack of interoperable information systems is a major one. The healthcare IT world has embraced SOA and many healthcare enterprises are currently integrating systems on SOA-based platforms. Despite these efforts, one of the most attractive features of SOA is scarcely reported on in the scientific community, namely the ability to reuse services and get the demonstrated benefits of reuse (Rine,Nada 2000).

National governments and international unions have published reports and initiated large research initiatives for social inclusion and independent living. Interoperable and cost-effective IT systems play an important role in achieving these goals. Standards development organizations (SDOs) working on interoperability in this domain are gradually starting to address component and service reuse, but as of yet there is to our knowledge no initiatives addressing reusable software service designs for the care and welfare of elderly. There seems to be a lack of knowledge about how SOA-based software services could be reused to provide social inclusion and independent living with an underlying concern for continuity of care.

From the software engineering perspective, reuse and interoperability are two key objectives of MDD. Hence, MDD should provide utility to developers of health and welfare information systems. UML tools are broadly applied in industry, but only as an exception for model-driven software development (Hutchinson et al. 2011). Some industries such as aviation have successfully extended UML with UML profiles having domain specific element to better support their needs (Fuhrmann et al. 2006; Schulte 2005). ***This has not yet been demonstrated for healthcare or care in general. There seems to be an opportunity for healthcare to explore using UML profiles to improve the usefulness of modelling service-based software systems.***

In general, there are few rigorous evaluations on using MDD. Some scientific articles report on positive experiences from using model-driven techniques in the healthcare domain (Blobel,Pharow 2005; Rubin et al. 2005; Jones et al. 2005; Raistrick 2005; Kawamoto,Lobach 2007; Omar 2006), but the adoption of MDD in healthcare software development is still low. As reported in (Mohagheghi,Dehlen 2008) the evaluation of the effects of using MDD is scarce and more research should be conducted to gain knowledge that could be used to improve MDD and software engineering in general.

This thesis addresses the gap between the development of healthcare standards and software services on the one hand and tailoring UML-based MDD tools on the other hand.

1.2 Research problem and questions

The overall research problem investigated in this thesis addresses the way IT systems facilitating continuity of care and independent living are developed.

There is a lack of knowledge about how model-driven development can assist in developing reusable software services that support continuity of care.

This places the research problem in the healthcare informatics field, with a strong focus on the informatics part and software engineering.

Continuity of care is considered an overarching concern for all healthcare information systems implementations. Focusing on the independent living for the elderly, the overall goal of providing continuity of care will depend on the ability to extend the information sharing from the traditional healthcare settings as general practitioners' offices and hospitals, and into the homes of the elderly, In this perspective, the research problem addresses the formal and informal care providers' need for information and management continuity. Many different care providers are involved in the care and management process, each with special access rights and information needs. The software services should be carefully designed and validated

In an *informatics* or *software engineering* perspective, the research problem definition addresses how MDD tools can be tailored to assist developers in creating software that can be reused in the domain. Reuse depends strongly on the services' interoperability qualities, adherence to domain standards, and availability of a clear design and documentation (Karlsson 1995) (chapter 7). Today, software developers in the healthcare domain have no or limited tool support for creating software that conform to relevant information structures and recommended architectural styles for continuity of care. This leads to a research problem statement:

How can software developers utilize model-driven development to develop reusable software services to support care and management of elderly in a homecare environment?

To narrow the focus of the investigation, the research problem addressed by this thesis can be summarized by the following research questions:

R1. How can a model-driven development toolchain with domain support aid the development of reusable domain software services?
R2. How can relevant domain standards and knowledge be incorporated into a model-driven development toolchain and what aid can they provide in the design and development process?
R3. Which reusable software services are relevant in the care and management of elderly living in their homes?

1.3 Research Approach

The research questions address practical challenges in both the healthcare and software engineering research area. The research approach must establish a clear

understanding of the actual needs in the target application environment and use best practices and methods from the two domains.

To investigate the research questions, two core artefacts are developed:

1. **A model-driven development toolchain** called the "ModelHealth Toolchain" that incorporates domain knowledge and provide assistance to the software developers in creating software services.
2. A **set of reusable software services** that support the needs of the domain – designed using the ModelHealth toolchain

To create these artefacts rigorously, a design science (Hevner et al. 2004) approach is applied. The approach has a strong focus on artefact creation and assessment, and emphasizes the importance of strong relationships to the target application environment and domain's knowledge base. An imperative feature of the design science research framework as defined by Hevner et al. is that an artefact may need to be refined in one or more assessment cycles.

To illustrate the relationships between the artefact research, relevance to continuity of care information systems, and the knowledge base, Figure 1 shows the applied research framework based on Hevner et al.'s "Information System Research Framework" in (Hevner et al. 2004). The framework specifies three important "cycles" (Hevner 2007) – *relevance (grey arrows), rigor (white arrows) and design cycles (black arrows).*

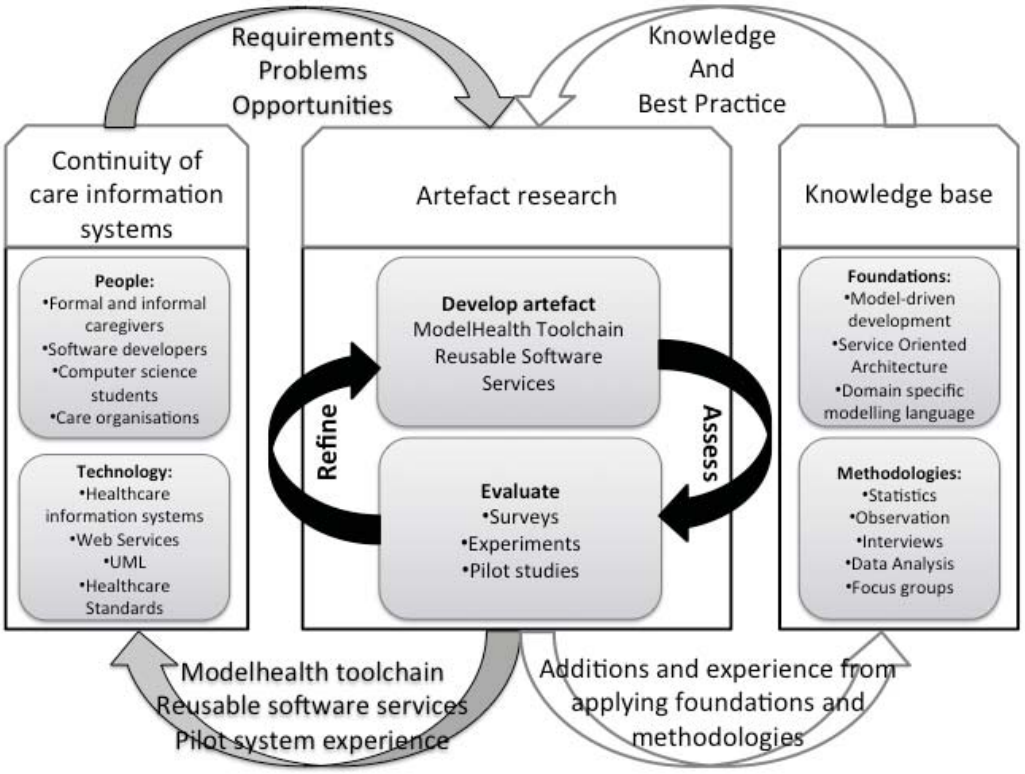

Figure 1: The applied research framework based on Hevner's IS research framework

- **Relevance Cycle** (grey arrows): The investigations in this thesis addressed the relevance cycle through a close relationship with the people, organizations and technology that are involved in the use and development of continuity of care information systems. Requirements, problems and opportunities were applied in creating the artefacts in the design cycle. A feedback loop was added from the design cycle where the results (experience and artefacts) were fed back to the environment for utilization and exploitation. The overall goal was to design and implement reusable domain software services and to develop a model-driven development toolchain (ModelHealth Toolchain) that could be applied by software developers (students and professionals) to create software services facilitating continuity of care.

- **Design Cycle** (black arrows): The design cycle is about creating, assessing and refining the core artefacts:

    - The ModelHealth Toolchain artefact was developed iteratively based on requirements from professional developers, best practice MDD, SOA design, and domain specific modelling language development. Within each design cycle, an assessment using rigorous evaluation methods initiated a refinement of the toolchain.

    - The software services were designed and developed based on requirement from relevant caregivers in the domain, best practice design methods. The evaluation of these services involved a relevant domain scenario with real end users.

    - The results from the design cycle were the artefacts and experiences acquired during the design cycle, and were communicated to the domain and knowledge base through scientific publications and reusable software artefacts.

- **Rigor Cycle** (white arrows): The rigor cycle entails the use of applicable foundations and methodologies from the "Knowledge Base". This can be sound theories, models, methods, best practices, and analysis techniques for the research domain.

    - Existing knowledge about the foundations of model-driven development, service oriented architectures, relevant standards for continuity of care, and domain best practices, were important input to the design of software services and the ModelHealth toolchain.

    - The artefact assessment was conducted applying best practice scientific methodologies.

    - Experiences, additions and nuances of foundations and methods from artefact design and evaluation, were fed back to "Knowledge Base".

Details about the development of the ModelHealth Toolchain artefact and the evaluation process and methods are described in detail in chapter Chapter 3 Research Method and Design

1.4 Research Context

This doctoral research project was initiated through the MPOWER project (MPOWER Consortium 2008b) that launched in October 2006. The main objective of the MPOWER project was to "*define and implement an open platform to simplify and speed up the task of developing and deploying services for persons with cognitive disabilities and elderly*." Working as the technical manager as well as leader for all the work on systems architecture and development approach, I had the possibility to influence the methods and directions of the project. My role in the core project activities is described in section 3.5.2. The MPOWER project successfully finished in June 2009.

Upon completing of the MPOWER project, I found it necessary to further investigate the research problems, and more investigations were conducted at the University of Tromsø. Additionally, I was given the opportunity to do a survey at a developer conference organized by the largest electronic health record developer in Norway, DIPS ASA. The thesis was finalized when I was working as a researcher in the universAAL project on developer tools and evaluations.

The work was conducted without a research scholarship. However, I was enrolled as a PhD student at the department of Computer Science at the University of Tromsø, Norway.

1.5 Claimed contributions and included papers

1.5.1 Contribution of thesis

The main objective of the thesis is to investigate how model-driven development extended with healthcare components can aid developers in creating reusable domain software services. The work has been carried out in a design-science framework and I claim that the thesis provides contributions in two areas:

- **Model-driven development toolchain design for the healthcare domain**: The research has produced guidelines for domain knowledge incorporation into model-driven development toolchains. The experience from building a DSML for a large domain with many stakeholders and domain assets is documented in a scientific paper. The key findings in the investigations strengthen the understanding of utilities of MDD. The utilities identified are development process support, traceability, transformation of code and documentation.
- **Reusable software services for continuity of care**: The research has produced reusable designs and implementations of software services from rigorous user scenario descriptions. The services are available open source through Source Forge. A SOA-application for continuity of care have been developed and evaluated in real life settings: A pilot study over one year found that relatively simple services can provide improvements in activities of daily living

The contributions can be summarized by a list of key findings from the investigations. Table 1 presents the eight key findings grouped into the two main contribution categories.

Table 1: List of key findings and the papers that address these findings

#	Finding	Addressed in paper(s)	Research Question(s)
F1	Continuity of care standard concepts relevant for service design can be modeled as UML Profiles	P1, P2, P4, P6, P10	R2
F2	Ease of use and correct code generation is important for the usefulness of MDD tools	P3, P10	R1
F3	Traceability services are considered an important utility in healthcare software development and can be provided using basic UML dependencies or using more sophisticated trace models	P3, P5, P8, P10	R1
F4	The toolchain should provide project structure and process assistance.	P3, P5, P10	R1
F5	Presentation of domain information in the design tools should be flexible, consistent and easy to use	P10	R2
F6	The modeling tool should provide design model verification and validation	P4, P10	R1
F7	A relatively small number of reusable software services cover a large part of the ICT support needs for independent living	P6, P9	R3
F8	Simple service-based applications have the potential to support older people at home, particularly older people with memory problems who need support in structuring the day and keeping an overview of the daily activities and appointments	P7, P9	R3

1.5.2 Included papers

Ten papers are included in this thesis as presented in Table 2.

Table 2 List of included papers

#	Paper title and forum
P1	"Improving systems interoperability with model-driven software development for healthcare", MEDINFO (Walderhaug, Mikalsen, Hartvigsen, Stav and Aagedal 2007)
P2	"The MPOWER Tool Chain - Enabling Rapid Development of Standards-based and Interoperable HomeCare Applications", Norwegian Informatics Conference (Walderhaug, Stav, Mikalsen and Jurisic 2007)
P3	"Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project", C2M workshop at ECMDA (Walderhaug, Mikalsen, Benc, Loniewski

	and Stav 2008)
P4	"Experiences from model-driven development of homecare services: UML profiles and domain models", MOTHIS workshop at MODELS (Walderhaug, Stav and Mikalsen 2008)
P5	"Traceability in Model-driven Software Development", in book chapter in Designing Software-Intensive Software, IGI (Walderhaug, Stav, Johansen and Olsen 2008)
P6	"Reusing models of actors and services in smart homecare to improve sustainability", MIE (Walderhaug, Stav and Mikalsen 2008)
P7	"Older people with and without dementia participating in the development of an individual plan with digital calendar and message board", Journal of Assistive technology, (Holthe and Walderhaug 2009)
P8	"Model-driven traceability in healthcare information systems development", MEDINFO (Walderhaug, Hartvigsen and Stav 2010)
P9	"Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned." International Journal of Medical Informatics, (Stav, Walderhaug, Mikalsen, Hanke and Benc 2011)
P10	"Evaluation of a Model-Driven Development Toolchain for Healthcare". Submitted to Automated Software Engineering, (Walderhaug 2012)

The relevance to this thesis and my contribution to each paper are described next.

P1: **Walderhaug, S**., Mikalsen, M., Hartvigsen, G., Stav, E., Aagedal, J.: *Improving systems interoperability with model-driven software development for healthcare*. Stud Health Technol Inform 129(Pt 1), 122-126 (2007)

Relevance to this thesis: The objective of this paper was to introduce the ModelHealth toolchain concepts and the mechanisms for incorporating healthcare knowledge into software developer tools. The paper presents three assertions addressing the overall problem of improving interoperability in healthcare information systems. These assertions are: model-driven development will improve interoperability between healthcare information systems, healthcare information standards are appropriate as reusable model-driven development artefacts, and healthcare information services in the homecare domain can be reused across organizations. These assertions outline the overall focus in the thesis.

My contribution: I wrote the problem definition and designed the toolchain with the running example. I wrote the paper with useful comments from the co-authors. I presented the paper at the MEDINFO 2007 conference in Brisbane, Australia.

P2: **Walderhaug, S**., Stav, E., Mikalsen, M.: *The MPOWER Tool Chain - Enabling Rapid Development of Standards-based and Interoperable HomeCare Applications*. In: Sandnes, F.E. (ed.) Norsk Informatikk Konferanse (NIK), Oslo, October 2007 2007, pp. 103-107. TAPIR (2007)

Relevance to this thesis: This paper presents the scope and design of the first version of the model-driven development toolchain. The main result presented in this paper is the selection and configuration of three core toolchain components: Sparx Enterprise Architect, NetBeans and SUN Application Server.

My contribution: I was responsible for the design of the toolchain and led the work on technology selection. Acting as the technical manager and responsible for development approach in the MPOWER project, I conducted the initial toolchain testing and developed training material for the toolchain within the MPOWER project. I wrote the paper with useful comments from the co-authors. I presented the paper as a poster at the Norwegian Informatics Conference in 2007.

P3: **Walderhaug, S.,** Mikalsen, M., Benc, I., Loniewski, G., Stav, E.: *Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project*. In: Bailey, T. (ed.) From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture, Berlin, Germany 2008. European Software Institute

Relevance to this thesis: This paper presents the evaluation of the toolchain done in the context of the MPOWER project. 16 developers from four European countries used the toolchain for one year, developing service designs from domain use cases and features. The developers took part in a survey addressing the ease of use, usefulness and work compatibility of the toolchain. The main findings were that perceived usefulness and ease of use are the most important factors for using a model-driven development approach. Specific comments from the developers on traceability and the correctness of the generated code resulted in a refinement of the toolchain.

My contribution: I was responsible for the design and conduction of the survey together with Marius Mikalsen. I analysed the data in SPSS and wrote the paper. Marius Mikalsen did data analysis review and provided useful comments along with the other co-authors. I presented the paper at the "From code-centric to model-centric software development" workshop at ECMDA-FA 2008 in Berlin, Germany.

P4: **Walderhaug, S.,** Stav, E.: *Experiences from model-driven development of homecare services: UML profiles and domain models*. Paper presented at the 2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems (MOTHIS 2008) in Toulouse, France.

Relevance to this thesis: This paper presents the approach to designing the domain specific modelling language as UML profiles for the target domain, homecare. The paper focuses on *which domain knowledge* that should be included into the language and *how this can be utilized* in the development phases. The paper outlines three steps to create two UML profiles within the MPOWER project. The UML profiles are: Homecare UML profile and Homecare SOA Profile. The paper was selected as best paper for the MOTHIS workshop at the MODELS 2008 conference.

My contribution: The work was done within the context of the MPOWER project. Whereas the information model design in MPOWER was a joint project effort, I did the UML Profile design having important discussions with Dr Erlend Stav. Acting as the technical manager and responsible for the model-driven development approach, I did the information modelling in Enterprise Architect. I wrote the paper with useful discussions with Dr Erlend Stav and Marius Mikalsen. I presented the paper at the MOTHIS workshop at the MODELS conference in Toulouse, France. ***The paper was selected as the session's best paper***.

P5: **Walderhaug, S**., Stav, E., Johansen, U., Olsen, G.K.: *Traceability in Model-driven Software Development*. In: Tiako, P.F. (ed.) Designing Software-Intensive Systems - Methods and Principles. pp. 133-160. IGI Global, Information Science Reference, Hersey, PA (2008)

Relevance to this thesis: This book chapter describes the foundation and core mechanisms of a central utility of model-driven development, namely traceability. One of the early findings in the toolchain evaluation was that traceability was considered a useful during development. This book chapter outlines four core traceability services: trace navigation, orphan analysis, coverage analysis and change impact analysis. The definition of the metamodel for representing trace information in the traceability services was necessary for the work on traceability in the ModelHealth toolchain.

My contribution: The work was carried out as a part of the EU project ModelWare. I was managing the work on traceability, and responsible for coordinating the effort with the other tasks in the project. The work was primarily carried out by Ulrik Johansen, Dr Erlend Stav and myself. The book chapter was written primarily by Ulrik Johansen, Erlend Stav and myself, with input on MOFScript from Gøran Olsen.

P6: **Walderhaug, S.,** Stav, E., Mikalsen, M.: *Reusing models of actors and services in smart homecare to improve sustainability*. Stud Health Technol Inform 136, 107-112 (2008)

Relevance to this thesis: This paper describes the process of creating the domain actor library to be used by the domain specific modelling language. Furthermore, it presents the set of reusable services that was implemented for the proof-of-concept applications reported in P7 and P9. The paper concludes that reusable model elements may reduce the gap between business processes and IT system realization.

My contribution: I was in charge of the specification of the domain actor library and carried out the harmonization with standards. Service specification was done as a long-term process in the MPOWER project, and acting as the technical manager and responsible for development approach I managed and contributed to this process. I wrote the paper with useful comments from Dr Erlend Stav and Marius Mikalsen, and presented the paper at the MIE 2008 conference in Gothenburg, Sweden.

P7: Holthe, T., **Walderhaug, S**.: *Older people with and without dementia participating in the development of an individual plan with digital calendar and message board*. Journal of Assistive Technologies 4(2), 15-25 (2010)

Relevance to this thesis: This paper reports from the 14-month pilot trial of the individual plan system developed in the MPOWER project. The evaluation shows that the underlying services identified in P6 can be combined into an useful support system for older people with cognitive impairments. Furthermore, the paper also points to some network communication challenges related to SOA as a concept for application deployment. The paper concludes that this kind of system has a great potential for future health and social care services.

My contribution: Torhild Holthe and her team handled the primary recruitment and contact with the elderly. I attended one home visit to a user and took part in the evaluation session. For all technical assistance and problem solving in the installation and trial period, I was the primary contact and responsible for communicating with the system developers. Acting as the technical manager in the MPOWER project, I was strongly involved in the system design and development process. I co-authored the paper with responsibility for the technical parts of the paper.

P8: **Walderhaug, S.,** Hartvigsen, G., Stav, E.: *Model-driven traceability in healthcare information systems development*. Stud Health Technol Inform 160(Pt 1), 242-246 (2010)

Relevance to this thesis: This paper reports from the experience developing two proof-of-concept systems in the MPOWER project in terms of model-driven traceability. The paper demonstrates how the core traceability services as defined in P5 can be implemented using the toolchain described in P3, reusable actors in P6 and the profiles described in P4. The paper concludes that traceability is both desired from a validation point of view (e.g. FDA guidelines for software validation) and user utility point of view (as reported in P3 and P10). Model-driven development allows for easy implementation of the core traceability services.

My contribution: As the principal author of the paper and the creator of the model-driven development toolchain, I did all the work on modelling and concept creation. I wrote the paper with useful comments from Professor Gunnar Hartvigsen and Dr Erlend Stav. Professor Gunnar Hartvigsen presented the paper at the MEDINFO 2010 conference in Cape Town, South Africa.

P9: Stav, E., **Walderhaug, S**., Mikalsen, M., Hanke, S., Benc, I.: Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned. International Journal of Medical Informatics (2011). doi:10.1016/j.ijmedinf.2011.03.007

Relevance to this thesis: This paper summarizes the development and evaluation of two SOA-based proof-of-concept applications in the MPOWER project. The main research questions addressed are how developers perceive

the use of model-driven development and SOA for application development in this domain, what are the reusable software services and are these sufficient to build sustainable support system for the domain. The main experience is that the chosen approach was found productive in developing information centric homecare applications. The paper provides a detailed view on the complete process from requirements elicitation to final evaluation of the realized systems.

My contribution: I was strongly involved in all phases of the design, development and evaluation of the development methods and system design. I was responsible for and contributed to the formalization of user needs in UML and the process of using these in the succeeding service identification and design steps. I was in charge of the SOA evaluation with the developers as part of the work presented in P3, and did the data analysis. I co-authored the majority of the sections of the paper together with Dr Erlend Stav and Marius Mikalsen. Input on medical and social information services as well as the Norwegian POCA design was provided by Dr Ivan Benc, whereas Sten Hanke wrote the communication services part.

P10: **Walderhaug, S.:** *Evaluation of a Model-Driven Development Toolchain for Healthcare*. Automated Software Engineering Journal, revision submitted on September 1st, 2012

Relevance to this thesis: This paper presents the empirical developer investigations done in this doctoral project. It shows the evolution of the ModelHealth toolchain in terms of a design cycle approach. The paper addresses to which extent model-driven development with domain support aid the developer in creating SOA-based healthcare information systems that conform to interoperability standards in the domain. The main results are that reusable domain modelling elements are useful for creating correct and complete designs and that traceability and model transformation are important utilities of MDD. Based on the empirical results, a set for guidelines for how to incorporate domain knowledge into model-driven development toolchains is provided.

My contribution: The work presented in the paper was carried out by me in the period from 2007 to 2011. I designed the overall research process (design cycle), designed and implemented the toolchain and transformations, designed and conducted the evaluations, and analysed the results. Dr Erlend Stav gave invaluable input to the technical toolchain solution, especially on the MOFScript transformation, and did an extensive review of the paper prior to submission. Marius Mikalsen provided feedback on the data analysis on the final student experiment.

1.6 Thesis structure

The remainder of this thesis consists of two parts.

PART I - Summary of the research process

Chapter	Content
2 – Background	This section introduces the problem domain and focus the scope of investigations. The core domain and technological concepts are presented with current state-of-the-art and challenges.
3 – Research Method and Design	In this section, the design science approach is introduced and its "instantiation" to this project is described in three phases. A large subsection is dedicated to the description of the ModelHealth toolchain evolution through the three phases. The final subsection presents an overview of the evaluation methods applied.
4 – Results	The results are organized according to the three research questions. For each question, the findings addressing the question are presented together with the supporting results from the individual papers.
5 – Discussion	The three research questions are discussed in terms of the findings, results and domain trends. A separate section is reserved for the experience from applying the design science approach. Finally, the chapter discusses implications for research and practices, limitations and recommendation for future research.
6 – Conclusions	Concludes the work based results and discussions of the research questions.

PART II - Included publications

P1. Walderhaug, S., Mikalsen, M., Hartvigsen, G., Stav, E., Aagedal, J.: Improving systems interoperability with model-driven software development for healthcare. Stud Health Technol Inform **129**(Pt 1), 122-126 (2007).

P2. Walderhaug, S., Stav, E., Mikalsen, M.: The MPOWER Tool Chain - Enabling Rapid Development of Standards-based and Interoperable HomeCare Applications. In: Sandnes, F.E. (ed.) Norsk Informatikk Konferanse (NIK), Oslo, October 2007 2007, pp. 103-107. TAPIR (2007).

P3. Walderhaug, S., Mikalsen, M., Benc, I., Loniewski, G., Stav, E.: Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project. In: Bailey, T. (ed.) From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture, Berlin, Germany 2008. European Software Institute.

P4. Walderhaug, S., Stav, E.: Experiences from model-driven development of homecare services: UML profiles and domain models. Paper presented at the 2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems (MOTHIS 2008), Toulouse, France. Selected as session's best paper.

P5. Walderhaug, S., Stav, E., Johansen, U., Olsen, G.K.: Traceability in Model-driven Software Development. In: Tiako, P.F. (ed.) Designing Software-Intensive Systems - Methods and Principles. pp. 133-160. IGI Global, Information Science Reference, Hersey, PA (2008)

P6. Walderhaug, S., Stav, E., Mikalsen, M.: Reusing models of actors and services in smart homecare to improve sustainability. Stud Health Technol Inform 136, 107-112 (2008)

P7. Holthe, T., Walderhaug, S.: Older people with and without dementia participating in the development of an individual plan with digital calendar and message board. Journal of Assistive Technologies ›(2), 15-25 (2010)

P8. Walderhaug, S., Hartvigsen, G., Stav, E.: Model-driven traceability in healthcare information systems development. Stud Health Technol Inform 160(Pt 1), 242-246 (2010).

P9. Stav, E., Walderhaug, S., Mikalsen, M., Hanke, S., Benc, I.: Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned. International Journal of Medical Informatics (2011). doi:10.1016/j.ijmedinf.2011.03.007

P10. Walderhaug, S.: Evaluation of a Model-Driven Development Toolchain for Healthcare. Automated Software Engineering Journal, revision submitted on September 1st, 2012.

Chapter 2 Background

This chapter describes the core concepts address in the doctoral project. As the included papers provide an extensive background description, the main purpose of this chapter is to present a holistic view and complement the paper descriptions where necessary.

The overall problem concepts are presented as shown in Figure 2. The figure shows an overview of how concepts in the healthcare domain are related to software engineering through the concept of interoperability and domain specific languages.

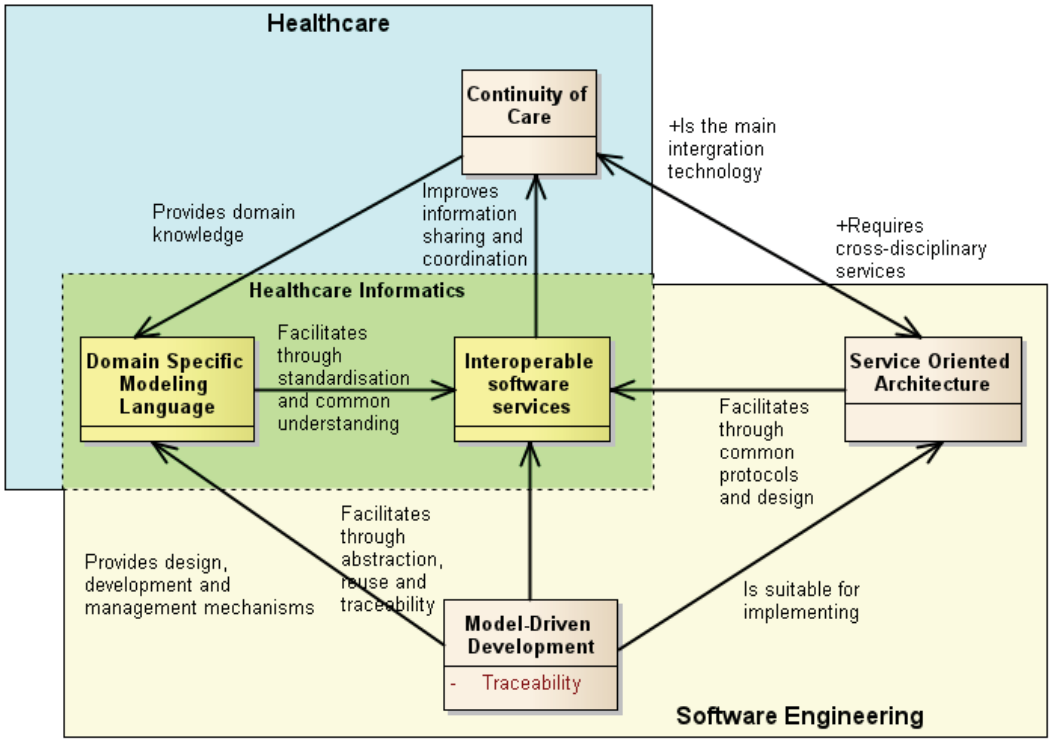

Figure 2 Overview of the core concepts and relationships in the research domain.

The focus of the research falls into the healthcare informatics area, focusing on how to construct a domain specific modeling language that can facilitate the development of interoperable software services. In the following, the core relationships and concepts are presented.

2.1 Independent living and continuity of care

Independent living is a goal for most people, and also for the care providers. The consumers of care services, typically elderly, chronically ill and cognitive disabled are empowered by state-of the art information and communication technology in their homes to achieve the overall goal of aging in place (Demiris et al. 2004; Magnusson et al. 2004; Wancata et al. 2003). An important challenges related to independent living is to support "continuity of care" for the users, defined by Haggerty et al. as the

"… degree to which a series of discrete healthcare events is experienced as coherent and connected and consistent with the patient's medical needs and personal context. Continuity of care is distinguished from other attributes of care by two core elements—care over time and the focus on individual patients." (Haggerty et al. 2003).

From a system architecture point of view, having a clear understanding of the stakeholders involved and their concerns, is of utmost importance for designing a sound system. The IEEE 1471-2000 "Recommended Practice for Architectural Description of Software-Intensive Systems" standard describes that *"concerns are those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders."* Having a clear understanding of this *"lay a foundation for quality and cost gains through standardization of elements and practices for architectural description."(IEEE 2000)*

In terms of healthcare and continuity of care, the TC 251 group in CEN[4] has developed the EN-13940-1: 2005 standard (also called CONTSYS) that defines the core concepts and stakeholders related to Continuity of Care using a UML class diagram notation (CEN TC251 2006). ISO TC215 is now developing the CONTSYS standard to become an ISO standard that will "*cover the generic concepts needed to achieve continuity of care."* and provide *"a clear conceptual framework to establish the terms of reference of health information systems. The system of concept as well as the process and workflow descriptions are meant as tools for the development of information systems."(ISO TC 215 under development).*

To illustrate how the CONTSYS standard defines a concept, Figure 3 shows the Subject of Care with its relationships to other healthcare concepts that may be relevant for the design of a software service to be used in e.g. systems integration.

[4] CEN TC 251 website: http://www.centc251.org

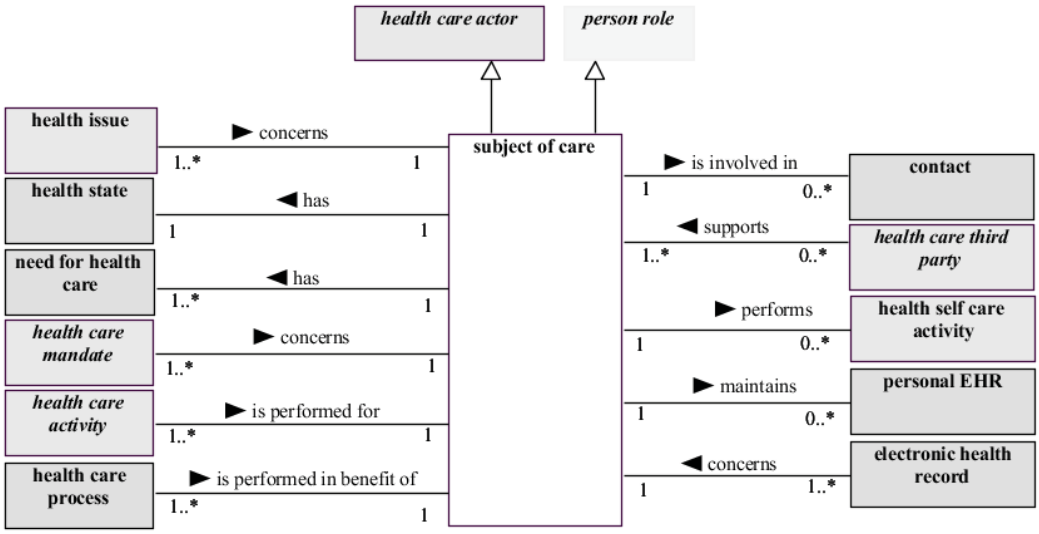

Figure 3 The Subject of Care concept from ISO 13940 (ISO TC 215 under development)

New technological solutions may facilitate independent living for older frail people and support carers taking care of people with dementia (Hagen et al. 2005). Increased attention has been given to both the role and potential of new technological solutions to support frail older people and their carers at home, as well as supporting health care staff in improving optimal use of time at work (Magnusson et al. 2004). Many technological solutions are present today, however, there is little knowledge about how to utilize this technology for older people as well as for people with dementia (ibid.). Few products are based upon involving these user groups in product development.

To create information systems that support the coordination of care across disciplines, interoperable and multi-disciplinary software services play an important role.

The technological solutions supporting independent living must operate in an existing IT environment in order to provide continuity of care. A main obstacle in achieving this is the lack of systems interoperability (Brailer 2005; Walker et al. 2005). New ways of providing continuity of care are being evaluated, based on teamwork treatment – demanding support from interoperable information systems. Interoperability in healthcare has been identified as an important area of research and development by many organizations, including the European Union (EU)[5], the Object Management Group (Object Management Group (OMG)) and other national organizations (Norwegian Ministry of Social Affairs,Norwegian Ministry of Health 2004). The ability to exchange information and share services across departmental, organizational and national borders can reduce the administrative overhead and costs (Walker et al. 2005), and as a result improve the effectiveness of healthcare provided. Consequently, more patients can be treated faster with the same amount of (care) resources.

[5] EU Life sciences, genomics and biotechnology for health website: http://cordis.europa.eu/lifescihealth/home.html

The demographic change in Europe requires require that we rethink traditional models of care. One prominent aspect of new care models used for chronic conditions is the cooperation of several different stakeholders (e.g., nurses, general practitioners, visiting nurses, the patient's relatives and the patient himself) in the care process (Winnem,Walderhaug 2002). These care models are often denoted continuity of care.

With continuity of care, patients are no longer passive consumers of health care services, but are instead demanding more and more control over their own treatment, together with increased responsiveness and improved quality of care services provided by the involved healthcare institutions. Today, healthcare systems are expected to maintain the continuity of care, shared care, and the empowerment of patients in the management process (Tattersall 2002).

The treatment and management of homecare consumers, typically elderly, chronically ill and cognitive disabled, require a coordinated effort from healthcare and social welfare services. To effectively support these care services with information systems, interoperability of core information such as patient care plan, calendar and medication-list is a prerequisite.

To improve interoperability between systems, the leading standardization bodies in healthcare information, HL7, CEN TC251 and OpenEHR, have specified standards that address systems architecture and information exchange. Although these standards have been available to the Health Information Systems (HIS) vendors for some time, the different HIS are not interoperable, requiring the development of software adapters to be able to exchange information about the patients. There is an urgent need for a standardized interface and method to realize this information exchange.

The new models of care, characterized by increasingly cooperating stakeholders and empowered users, would benefit from interoperable health information systems. The healthcare information systems are no longer stand-alone applications with a database, some specific business logic and a product-specific graphical user interface, but a distributed system of resource and functional services. To share information, different middleware services are used, including CORBA, Java RMI, DCOM or Message-oriented Middleware (MoM). Syntactic compatibility is achieved using messaging standards based on e.g., HL7 messages and EDIFact. However, this is not sufficient to benefit fully from an interoperable health information network. Guise and Kuhn says in (Giuse,Kuhn 2003), with references to (Bleich,Slack 1992; Stead 2000):

"Open architectures for HIS are not widespread; today's commercial systems seldom go beyond providing simple HL7 interfaces for data exchanges at the syntactical level. Many systems are still strongly tied to internal databases in a 'vertical stovepipe' model, and their data definitions are not transparent enough to support ready functional integration."

The standardization bodies provide limited tool support to the developers of health information systems. To incorporate standard healthcare concepts in the systems' design, an operational software engineering artefact that provides both semantic and syntactic interoperability functionality (Beale 2002; Park 2004) should be available for the system architects and developers (Kuhn et al. 2003; Lenz et al. 2007; Lenz,Kuhn 2004).

Interoperability has many definitions, and the definition of *working interoperability* from HL7 is considered appropriate for this thesis. In the Service-Aware Interoperability Framework (SAIF) [6] from Health Level 7 (HL7) *working interoperability* is defined as: *"The collection of structures, processes, and components that support Computable Semantic Interoperability (CSI) between two parties ("trading partners") who are interacting (for example, exchanging information, coordinating behavior) to achieve one or more business goals.* ***Interoperability****, in this context, is further defined to be the* ***deterministic*** *exchange of data or information in a manner that* ***preserves shared meaning****."*

Healthcare interoperability is addressed by the European Commission in "Commission recommendation of 2 July 2008 on cross-border interoperability of electronic health record systems" stating that: "*Lack of interoperability of electronic health record systems is one of the major obstacles for realising the social and economic benefits of eHealth in the Community. Market fragmentation in eHealth is aggravated by the lack of technical and semantic interoperability."(European Commision 2008)*. A recent survey among the EU member states revealed that there is huge variation between the European countries with respect to implementing the EU eHealth interoperability recommendations. The member states express a need for more guidance on interoperability **implementation** (Calliope Network 2008).

To aid the implementation of interoperable support systems, standards and reusable components may play an important role (Sametinger 1997). Standards developing organizations (SDOs) in the healthcare domain are working on standardization efforts that aim to provide information interoperability in healthcare (Eichelberg et al. 2005).

Some of the latest additions to the information sharing standards are those addressing the use of Service Oriented Architecture (SOA) (Erl 2006) in healthcare as a means to overcome interoperability and reuse challenges in the domain. The OMG/HL7 Healthcare Service Specification Project (HSSP Project 2007) and IHE (IHE 2012) have proposed architecture and methodology documents for designing service systems that adhere to the core principles and standards in the domain (Honey,Lund 2006; Honey et al. 2006).

2.2 Interoperability with Service Oriented Architectures in the Healthcare Domain

SOA specifies *"a paradigm for organizing and utilizing distributed* ***capabilities*** *that may be under the control of different ownership domains"* (OASIS Open 2006). The SOA paradigm reflects the way patient information is distributed in different systems owned by different organizations, and provides an architectural framework for sharing data and services. An example of a popular implementation of SOA is Web Services that specifies the main transport protocols and formats to use, focusing on web technologies such as XML, SOAP and HTTP (World Wide Web Consortium (W3C) 2004, 2007). From an interoperability viewpoint, technical interoperability is

[6] SAIF: http://wiki.hl7.org/index.php?title=Product_SAIF

achieved through the use of a Web Services-based integration approach. Information interoperability on the other hand, is about agreeing on basic domain concepts, and using this agreement correctly in the design and development of the information systems. The use of web services does not provide direct support for information interoperability.

The trend towards a service-oriented architecture (SOA) for healthcare information systems represents a new area of research in the healthcare informatics field. The domain is characterized by a large number of stakeholders having an interest in the system, and a plethora of existing information systems that should be interfaced (Grimson et al. 2000). SOA have become an important strategy to implementing interoperability in many domains, also the healthcare domain (Kawamoto,Lobach 2007). The Danish MEDCOM has developed a guide to "*The Good WebService*" where they provide a "profile" for how they would like to design web services in the Danish Health system[7]. This profile has information about protocols, information formats, error coding, security mechanisms and other design specific decisions. As the healthcare enterprises and welfare services are moving towards SOA based architecture, it will be of utmost importance that the system developers adhere to these profiles and reuse service designs wherever relevant.

One way to improve systems interoperability is to share service designs and reference implementations, including documentation of how a standard or guideline is interpreted and implemented. A SOA-based implementation of a healthcare system is evaluated in (Raghupathi,Kesh 2008). The authors present the evaluation of a prototype implementation of a SOA-based interoperable electronic health record (EHR), identifying the main design challenges. They conclude that SOA provides potential value to interoperable EHRs, but there are some challenges with SOA design: "*The health care industry particularly faces the challenges of incomplete standards (e.g., of medical terminology) and lack of robust development and modeling tools*". [ibid]

2.3 Model-Driven Development process, utilities and tools

The acronym for model-driven development, MDD (Mellor 2004), labels the development technique where models are applied as the main artifacts in the development process to create application code and corresponding documentation. The models are developed by creating diagrams with a graphical representation of the underlying language's elements. OMG's MDA® (Miller,Mukerji 2003) is a specific approach to doing MDD, whereas model-driven engineering (MDE) is considered a broader term. Model-driven software development (MDSD) as presented in (Stahl,Völter 2006) is a more narrow term than MDD with a strong focus on the application code creation. MDD is the main term used to describe model-driven development in this thesis.

MDD has been around for many years, in the context of computer assisted software engineering (CASE) tools in the early nineteen eighties (Iivari 1996; Lundell,Lings

[7] Danish Medcom – The good webservice: http://www.medcom.dk/wm110731

2004; Sharma,Rai 2000), and revitalized through the popularity of UML (Object Management Group (OMG) 2007) and OMG's MDA approach (Mellor 2004; Miller,Mukerji 2003). Miller and Mukerji states in the MDA Guide (Miller,Mukerji 2003): *"The three primary goals of MDA are portability,* ***interoperability and reusability*** *through architectural separation of concerns."* Recently, there has been an increased interest in software development using Model-Driven Development (MDD) techniques (Hailpern,Tarr 2006).

Adherence to design guidelines and profiles can be imposed by software developer tools. The majority of software developers use advanced tools in their work to assist the process of creating artefacts such as application code, design diagrams, application documentation and test reports. There is a potential for improving these tools to provide more contextual support based on the target environment or system platform.

In 2002, the Object Management Group (OMG) introduced the Model-Driven Architecture (MDA)(Miller,Mukerji 2003), an approach focusing on using models (e.g., UML models(Object Management Group (OMG) 2005)) as first-class entities in the development of software systems. In practice, this means that the models are used directly in the implementation of an information system, either as system blueprints or as input to code generation engines that produce executable code. MDA is the most known model-driven development (MDD) approach, and the overall idea is to separate business functions (in Platform Independent Models - PIM) from its technological implementations (in Platform Specific Models – PSM), enabling code generation and reuse of components. The overall benefit is improved interoperability and reduced development time and cost.

Using a MDD approach in the development of healthcare information system services could facilitate the use of standards through specification of reusable standards-based PIMs. Advanced UML mechanisms such as UML Profiles could be used to further extend the expressiveness of the modeling language and force the use of standardized healthcare concepts. As a result, the developed systems can increase the level of interoperability, and at the same time development and maintenance costs will decrease. The goals of MDD described in (Stahl,Völter 2006) can be summarized as follows:

- Increase development speed and software quality through automation
- Higher level of reusability as the architectures, modeling languages and transformations are generic for the domain (abstract)
- Improved manageability of complexity through abstraction
- MDSD is based on the Object Management Group's Model Driven Architecture ® (MDA). OMG's focus in on interoperability, portability and reusability through architectural separation of concerns (Object Management Group (OMG) 2003)

MDD seeks to use models (a formal graphical notation) to represent all artefacts involved in the development of a software system. Models are both abstract and formal at the same time, meaning that irrelevant details are abstracted away and the core is described (modeled) unambiguously. The models are used in diagrams that

specify a static or dynamic (behavior) view of the target system. Diagrams are typically created using a top-down approach where the high-level concepts are identified and documented before they are broken down into sub-concepts, workflows and information models. The low-level detailed models can be transformed into new and technology specific models (including concepts from J2EE or .Net). From the technology specific models, executable code can be generated.

2.3.1 OMG's Model Driven Architecture (MDA)

The best known model-driven development approach is OMG's Model Driven Architecture (MDA)(Mellor 2004; Miller,Mukerji 2003). MDA provides an open, vendor-neutral approach to the challenge of business and technology change. Based on OMG's established standards, the MDA separates business and application logic from underlying platform technology. The platform-independent models (PIM) of an application or integrated system's business functionality and behavior, built using UML and the other associated OMG Modeling standards, can be realized through MDA on virtually any platform, open or proprietary, including Web Services, .Net, CORBA, J2EE, and others. These platform-independent models document the business functionality and behavior of an application separate from the technology-specific code that implements it, enabling interoperability both within and across platform boundaries. No longer tied to each other, the business and technical aspects of an application or integrated system can each evolve at its own pace – business logic responding to business need, and technology taking advantage of new developments – as the business requires (Object Management Group (OMG) 2003).

One of the core features of MDA (and MDD) is the ability to transform one model of the system into a new technology specific model, which in turn can be used to generate executable code. The overall concept is to model the system from different viewpoints, each viewpoint having its own goal and role in the development process. MDA describes three different viewpoints and their corresponding models, namely the Computation Independent Model (CIM), Platform[8] Independent Model (PIM) and Platform Specific Model (PSM)(Object Management Group (OMG) 2003).

- CIM: The CIM is a model that focuses on the on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined.

- PIM: The platform independent viewpoint focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another.

[8] A *Platform* in MDA is defined as a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented

- PSM: The platform specific viewpoint combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

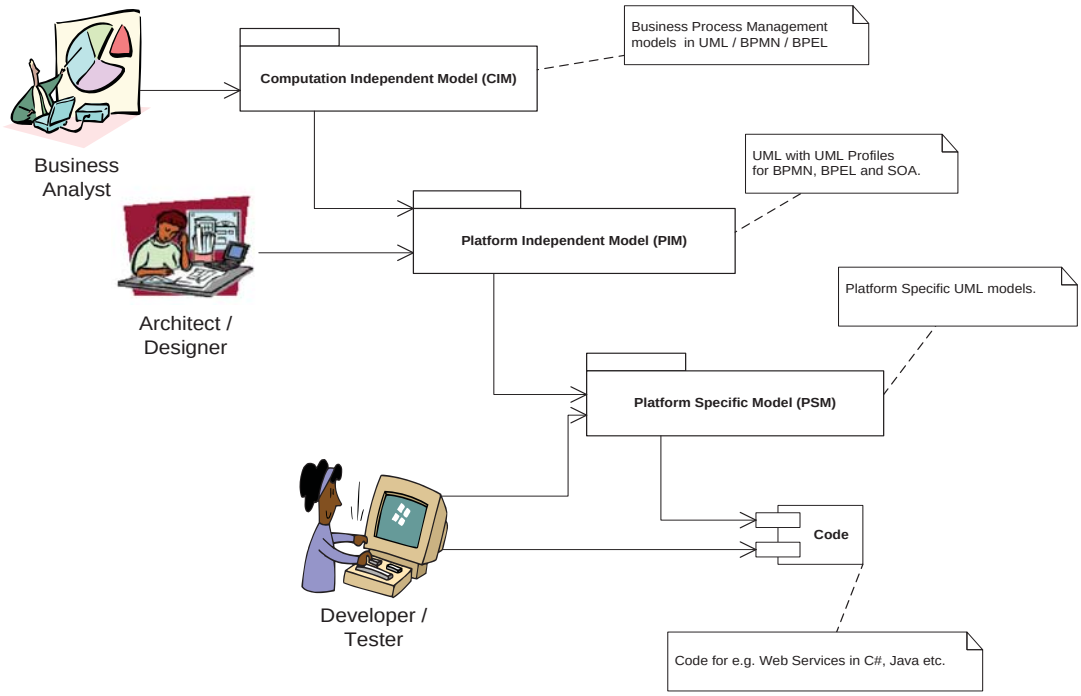

Figure 4: The MDA models. Figure based on (Rosen 2004)

MDD is a promising development approach where the development cost is reduced through extensive model reuse, model transformation and code generation techniques. In addition, the quality of the final executable code will be better due to reuse of verified code and software patterns. Many organizations have reported significant improvements in the development process and on software quality (Hartman 2006; Guttman,Parodi 2006), but the scientific evidence is still scarce (Mohagheghi,Dehlen 2008).

2.3.2 Domain Specific Modeling Languages and UML Profiles

A domain specific modeling language (DSML) incorporates modeling elements and design structures that are specific for a domain, and typically allows for more effective and precise modeling (Fowler 2010). DSMLs are being developed for many domains to improve the efficiency and usefulness of modeling in the development process (Kärnä et al. 2009; Tolvanen,Rossi 2003), the majority involving hardware interaction such as cell phone software and heart rate monitor watches. A DSML can take the form of a complete meta model or as an extension of an existing metamodel such as UML. The objective is to increase the expressiveness and thus be able execute the semantic models directly and improve the generated artefacts (code, documentation and scripts) (Fowler 2010). In this thesis, the terms DSML and DSL are used interchangeably. Domain specific modeling languages, reusable models and model-structures, model checking, transformations and traceability are the main mechanisms that can aid the developers in designing software systems that adhere to the design guidelines and profiles.

One way to achieve semantically and syntactical information interoperability is to have a common metamodel. A metamodel describes the semantics of a language, such as an information standard in healthcare, and must be described formally. In a model-driven development project one may use several metamodels, but to ensure interoperability one should have a mapping model between the different metamodels. A metamodel may be domain specific, e.g. a metamodel for continuity of care, providing a familiar language to the people modelling information systems for this domain.

OMG has specified the Unified Modeling language (Object Management Group (OMG) 2005), a formal modelling language that can be used to specify (model) both static and behavioral aspects of a system. Using the Object Constraint Language (OCL)(Object Management Group (OMG) 2006; Warmer 2003) together with UML enables the modeller to enrich the models with enough detail to render it possible to generate high-quality source code. Another mechanism offered by UML is the use of UML Profiles. A profile is used to add domain specific concepts and terms to the modeling language. Stereotypes, tagged values that can be applied to elements, attributes, associations etc., to enable accurate modeling for a specific domain.

In (Khambati et al. 2008a), Khambati et al presents a model-driven development of a prototype for creating care plans using a tailored DSML based on a meta language. The evaluations of the prototype show positive results, and especially for improved collaboration and reuse on care plans. Khambati et al states in (Khambati et al. 2008b) that *"An important direction for this sort of work is to achieve alignment with key international standards"*.

As described in Paper 10 (Walderhaug 2012), Khambati et al is uses a small DSL for care plans that has high "closeness of mapping" that was perceived positive in expert evaluations. In (Kosar et al. 2012), the authors report from a study where they compared the comprehension correctness and efficiency of students creating feature diagrams in a DSL and a GPL (Java). The results clearly showed that the DSL performed better than the GPL. Also in the study by Cao et al, the DSL/DSM solution performs better than plain UML with respect to correctness, comprehension and changeability (Cao et al. 2009). However, the construction of a DSL is resource demanding as shown in (van den Bos,van der Storm 2011; Fister Jr et al. 2011), which may be a barrier to creation and adoption of DSLs. Mernik et al discuss when and how a DSL should be developed in (Mernik et al. 2005). Even though most of the DSLs discussed are textual DSLs, many of the challenges are relevant for domain specific modelling languages, e.g. "*How can DSL development tools generated by language development systems and toolkits be integrated with other software development tools?*"

2.3.3 Traceability

Traceability, being a core aspect of MDD, has recently been subject to increased research. Winkler and Pilgrim presents a survey of traceability in MDD in (Winkler,Pilgrim 2009) where they conclude that: "*traceability methods are not used in practice as much as they could. One of the main reasons is lack of good tool support"* - and that more research should be conducted in a coordinated manner.

Increasing understanding and communication in the development team can benefit from a working traceability scheme (Limón,Garbajosa 2005). The ability to conduct traceability analyses is an imperative feature of MDD tools (Aizenbud-Reshef et al. 2006; Walderhaug et al. 2008b; Winkler,Pilgrim 2009) that can be defined as "*the ability to track any relationship that exists between artifacts involved in the software-engineering life cycle"(Walderhaug et al. 2008b)*. The importance of a clear traceability scheme for software development in healthcare is explicitly outlined by the Food and Drug Administration in (U.S. Department Of Health and Human Services et al. 2002) saying that: "*A traceability analysis should be conducted to verify that the software design implements all of the software requirements*" (page 19) and that "*a source code traceability analysis should be conducted and documented to verify that: each element of the software design specification has been implemented in code; modules and functions implemented in code can be traced back to an element in the software design specification and to the risk analysis*"(page 21).

In (Lago et al. 2009) the authors address the complexity of traceability in a "*scoped approach to traceability management"* for traditional software development. The approach reduces the number of links by identifying which tracelinks (trace paths) that are important for a type of system or an application domain. This reduces the complexity and management efforts required to maintain a complete and updated traceability repository. In MDD, the scoped approach may not be necessary as most tracelinks are implicit in the design models. Traceability is considered important by Hailpern and Tarr: *"the property of traceability (which enables creating or following a trace) is core to the value proposition of MDD" (Hailpern,Tarr 2006)*.

2.3.4 Quality of models

The success of MDD depends on the quality of models creating using the development approach.

A light-weight approach to evaluate the model quality is described as the 6C quality criteria presented in (Mohagheghi et al. 2009a) were used as basis. While the 6C model quality evaluation is rather simple, the SEQUAL framework presented in (Krogstie 2012) provides a complete evaluation framework for models where the quality is discussed on seven levels: physical, empirical, syntactic, semantic, pragmatic, social, and deontic. An important element of the framework is that it takes into account the "goal" of modelling which is a necessary inclusion because it is, as stated in the framework: *"For anything but extremely simple and highly intersubjectively agreed domains, total validity, completeness, comprehension and agreement as described above cannot be achieved"(ibid. chapter 4)*. The quality must be achieved in terms of the goals of modelling.

Table 3 presents a summary of the most relevant quality criteria to be used in model quality evaluation in the PhD project. As discussed in Paper 10 (Walderhaug 2012), five of the six criteria from 6C were deemed relevant for the final student experiment. As Krogstie's book was published after submission of Paper 10, the 6C criteria were used here. However, the criteria applied in the thesis corresponds to the more detailed criteria defined in the SEQUAL framework (ibid.). To put the applied 6C criteria (Mohagheghi et al. 2009a) in a broader perspective, the correspondence is shown in the "SEQUAL Label" column in Table 3.

Table 3 The model quality criteria applied for quality evaluation

6C Criteria	SEQUAL Quality Level	Details
Correctness	Perceived Semantic Validity	The right elements and their relationships should be included in the diagrams. Using the correct syntax, terms (naming conventions) from the domain are key factors to be evaluated. The syntactic quality as described in SEQUAL is ensured by the modelling tool in the majority of cases.
Completeness	Perceived Semantic Completeness	As the models are the basis for documentation and transformation, they need to be complete with respect to elements and properties.
Consistency	Social Quality	Defined as *no contradictions in the model*. It covers consistency between views or diagrams that belong to the same level of abstraction or development phase, and between models or diagrams that represent the same aspect, but at different levels of abstraction or in different development phases. In addition, it covers semantic consistency between models.
Comprehensibility	Pragmatic Quality - comprehension	How the *users* understand and select elements from the modelling language. SEQUAL express this as: "*the interpretation by human stakeholders of the model is correct relative to what is meant to be expressed in the model*." SEQUAL uses the term *"comprehensibility"* for the *model's* ability to be understood.
Confinement	Deontic Quality – feasible validity and completeness	A measurement of the level of abstraction and detail. The models should not have more details than necessary. This quality is related to both correctness and completeness.
Changeability	N/A	Defined as "supporting changes or improvements so that models can be changed or evolved rapidly and continuously".

2.3.5 Model Driven Development Tools

The core developer artefact in MDD is the MDD tool. The tool can be standalone or toolchain configured from different standalone tools that together comprise a MDD

tool. There is not a single definition of what constitutes a proper MDD tool. However, OMG maintains a list of MDA compliant tools on their webpage[9].

The availability, cost and quality of tools are considered crucial for MDD adoption in industry (Staron 2006). Unfortunately, MDD tools are part of the reason why MDD adoption is still low. The immaturity of current MDD tools has been identified as a problem in several empirical studies (Iivari 1996; MacDonald et al. 2005; Mohagheghi,Dehlen 2008; Staron 2006), and important tool requirements have been identified and specified by MacDonald et al. in (MacDonald et al. 2005) and Staron in (Staron 2006). The requirements include cost estimation, availability of rich libraries, support for domain knowledge, improve design quality by increasing understanding, support communication within development team, and provision of traceability throughout software development artefacts.

More generic requirement focusing on "Physical" qualities level is defined by Krogstie in (Krogstie 2012) where he presents a list of requirements in three different categories: model repository, model interchange and support for meta-modelling. These requirements were developed in the ATHENA A1 EU-project.

An industry assessment of how model driven engineering (MDE) was being applied and what are the success/failure factors was recently published by Hutchinson et al in (Hutchinson et al. 2011). From a survey among 250 MDD users (approx. 85% using UML and 40% using a DSL) and in-depth interview with 19 developers, they identified several interesting aspects.

- 66% think that MDD improves the communication between stakeholders (a quarter disagree)
- 47% of the respondents think that MDD will allow less experienced developers to do development (35% disagree).
- 74% think that MDD will require extra training (less than 9% disagree)
- 43% think UML is too complex (32% disagree)
- 46% think that MDD tools are too expensive (24% disagree).
- And most importantly, 56% think that organizations are using inappropriate MDD tools (12% disagree). From the interviews the authors report that *"Some users believe that had they adopted off-the-shelf tools, it would effectively have killed that adoption of MDE."*

The authors conclude that MDD is far from complete.

The VTT report on "Model-Driven Development: Processes and practices", the Parviainen et al report from a survey with 69 respondents from both academia and industry (Parviainen et al. 2009). The results show that about 50% say that the purpose of using UML is to generate code, whereas the other half uses UML mainly for documentation.

[9] OMG's list of MDA tools: http://www.omg.org/mda/committed-products.htm

There are many different tools available for the developers, both commercially and free of charge. There is no sound statistics showing which is the most used tool, and the VTT report shows that there is a large variance in the tools being used (see Figure 5.

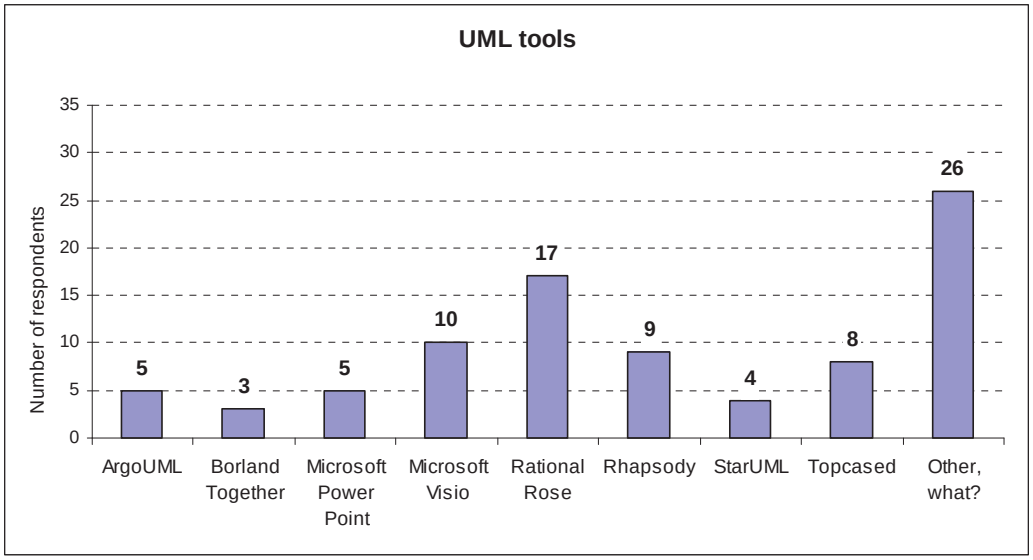

Figure 5 UML tools in organizations. From the MOSIS report (Parviainen et al. 2009).

In the "Other" category, includes *"MagicDraw, PoseidonUML, MetaEdit+, Papyrus, Omundo / Eclipse, Bouml, IBM Rational Software Modeler, Topcased, Eclipse UML2, Rational Software Architect, Eclipse UML2 Tools, GMF, RSA/RSM, ADONIS, Objecteering, Telelogic Tau, Netbeans, Enterprise Architect, and Sparx's Enterprise Architect*".

The VTT report also includes an analysis of the practices that is in accordance with the findings by Hutchinson (Hutchinson et al. 2011).

2.4 Empirical evaluations of model-based development approaches

Since the late eighties, CASE has been developed, evaluated and discussed passionately in the computer science / software engineering community. Many software companies invested lots of time and money in CASE technology without getting the desired return of investment. A common perception of CASE technology from the late nineties was that it failed to deliver upon its promises.

In (Lundell,Lings 2004), Lundell and Lings state that successful deployment CASE technologies are "at best variable", and they suggest that the expectations to MDD (or CASE tools in general) are unrealistic(Lundell,Lings 2004). In 1996, Iivari presented and article on CASE tools adoption where he reports that perceived complexity of tools has a negative effect of their usage, and that perceived tool effectiveness has a strong positive effect(Iivari 1996). In 2000, Sharma and Rai published an empirical investigation on CASE deployment in IS organizations where they report that CASE

tools are used on half of the development tasks in just a small subset of the development projects(Sharma,Rai 2000).

Despite significant investments and development efforts, the number of sound scientific evaluation publications of the qualities of model driven software development is low (Mohagheghi,Dehlen 2008). A few studies, like those conducted by MacDonald et al (MacDonald et al. 2005) and Arisholm et al (Arisholm et al. 2006) found no or just minor differences between using model-driven development techniques compared to traditional software development techniques. Others, like the Middleware Company' MDA productivity analysis (The Middleware Company 2003), found significant improvement with respect to productivity and code quality. A more generic MDD evaluation was done in the ModelWare project where Hartman reports *"significant gains of 20-60% were observed in the execution of maintenance task"* for some experiments, but also significant productivity losses mainly due to immature tools (Hartman 2006). The evaluation of MDD performance and utility results vary, and there seems to be a common agreement that *"model-driven engineering is still in its infancy*" and that there is still a need for more empirical studies in this field. (Mohagheghi,Dehlen 2008; Staron 2006).

Chapter 3 Research Method and Design

The introduction presented the adapted research framework applied in this thesis. This chapter describes each cycle in detail.

3.1 The Design Science approach

As presented in the introduction (section 1.3 and Figure 1), the research was organized in a design science framework, focusing on the development and assessment of two core artefacts, namely the ModelHealth toolchain and the set of reusable software services. Design science was considered appropriate for this doctoral project as it "*creates and evaluates IT artefacts intended to solve identified organizational problems. Such artefacts are represented in a structured form that may vary from software, formal logic, and rigorous mathematics to informal natural language descriptions. As field studies enable behavioural- science researchers to understand organizational phenomena in context, the process of constructing and exercising innovative IT artefacts enable design-science researchers to understand the problem addressed by the artefact and the feasibility of their approach to its solution"* (Nunamaker et al. 1991a).

Equally important as the artefact creation was the use of rigorous methods and relationship to the target application domain. The design science approach to finding relevant and rigor artefact solutions is built upon pertinent principles that are presented below:

- **Goal is to find utility – not truth**. (Hevner et al. 2004). "*Truth informs design and utility informs theory. An artefact may have utility because of some as yet undiscovered truth. A theory may het to be developed to the point where its truth can be incorporated into design.*" The investigation in this thesis seeks identify reusable software services for continuity of care that support independent living, and explore to which extent MDD can assist in developing reusable software service for the domain.
- **Design as a search process:** design science prescribes an iterative approach to artefact development with define/refine cycles. This approach is considered useful for the research conducted in this thesis as:
    - There is no single solution to the research problem addressed. The design science approach allows for an explorative investigation where

characteristics of both the users and the artefact can be evaluated. Creating reusable software services requires several iterations.

  - As presented in the introduction, there is no single model-driven development tool available that provides domain specific support for the defined target domain. A customized toolchain has to be developed and evaluated in several cycles. This toolchain will be a core design artefact along with the domain software services.

- **Relevance** plays a fundamental role in design science, thus the approach stimulates incorporation of the real needs of the end users / community. The reusable software services and the ModelHealth toolchain artefact address accepted challenges in the target application domain.

- Reusing domain knowledge in the **rigor** cycle: Design science emphasizes the importance of using knowledge from the "Knowledge base". Building upon best practice in the domain in terms of system integration approaches, continuity of care standards, foundations of MDD and evaluation techniques, is crucial for building a relevant and viable software services and tools for the specified target application domain.

The next section describes how the activities in these cycles were organized in three phases.

3.2 Three phases of design and evaluation

The artefact design process was split into three main phases as shown in Figure 6. The figure shows the Phase element with time period and the main objectives in the phase. Below each are the results from the process represented with artefact elements. Each phase is described in detail in the proceeding subsections.

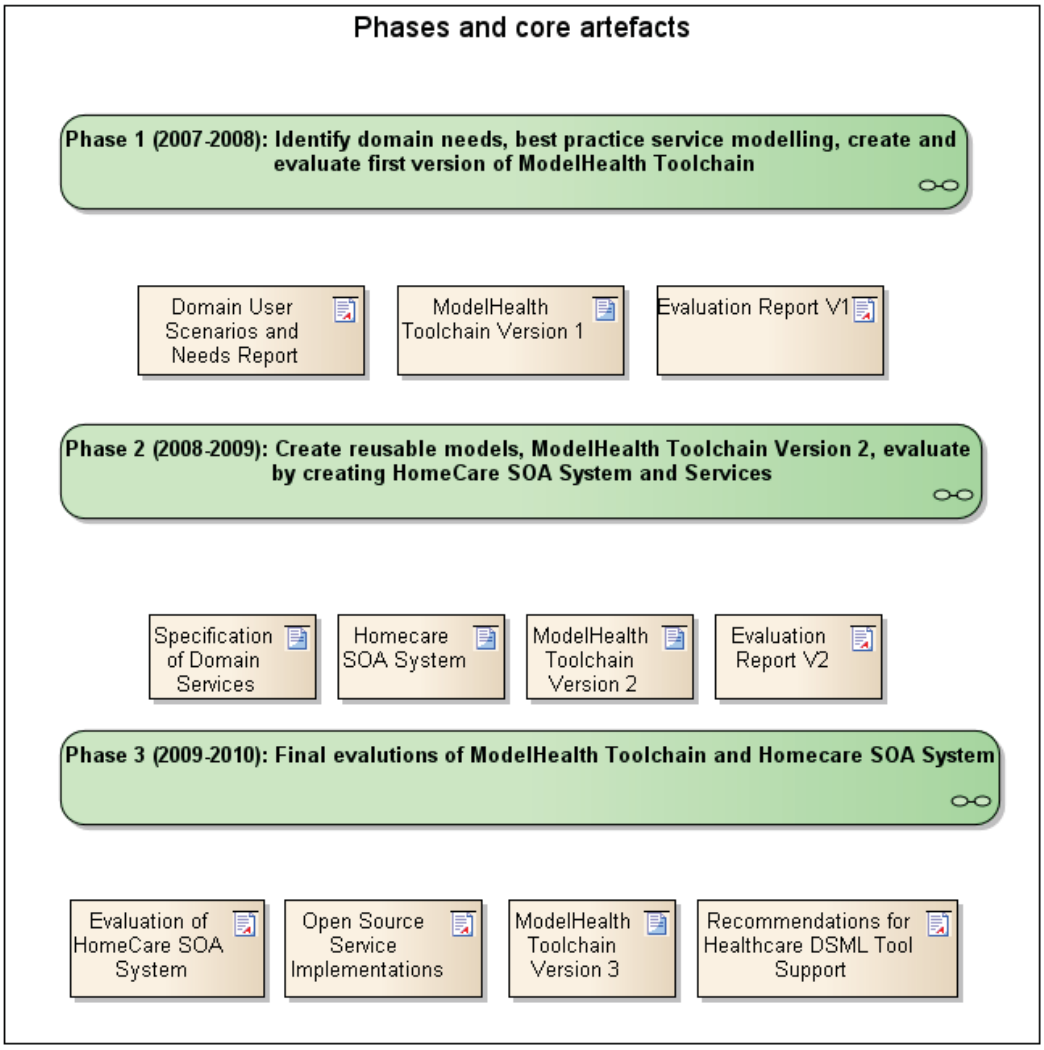

Figure 6 Diagram showing the three design phases and the results from these

3.2.1 Phase 1: Capture domain needs and best practice SOA and MDD

The main objective in phase 1 was to capture all relevant domain needs and state of the art SOA and MDD modelling. Based on this information, the first design cycle should be completed. Each activity and result is described in Table 4.

Table 4 The activities and results in Phase 1

Activity	Result	Comment
Identify domain actors and needs	Domain user scenario and needs report. The report presents typical everyday scenarios for the target users and outlines the required technology support. The report uses both textual descriptions and UML use cases.	To capture the needs for ICT-based assistive service in the domain and model the actor to use case relationships together with domain experts and representatives. The process is described in detail in Paper 6, 7 and 9.

Specify Initial ModelHealth Toolchain requirements	Initial toolchain requirements.	The requirements were specified by the technical partners in the MPOWER project. The specification and the process are described in section 3.3 and Paper 10.
Design and evaluate ModelHealth toolchain V1	ModelHealth Toolchain V1 Evaluation report	A first evaluation of version 1 of the ModelHealth toolchain. Details are presented in Paper 2 and 10.
Identity best practice MDD and SOA modelling	MDD Toolchain best practice	Survey existing tools and techniques in the field of MDD and SOA. Focus on open source / low-cost solutions and solutions based on standards from e.g. OMG.
Specify evaluation approach	Evaluation plan	Plan the evaluation of the toolchain and services/application. Use best practice methods within the limitations of the project. Recruitment of users and allocation of resources was found crucial. The evaluation plan for applications was made as a deliverable in the MPOWER project, whereas the evaluation plan is described in section 3.3 and Paper 10.

An illustration of the relationships between the activities and results in the three cycles of Phase 1 is presented in Figure 7.

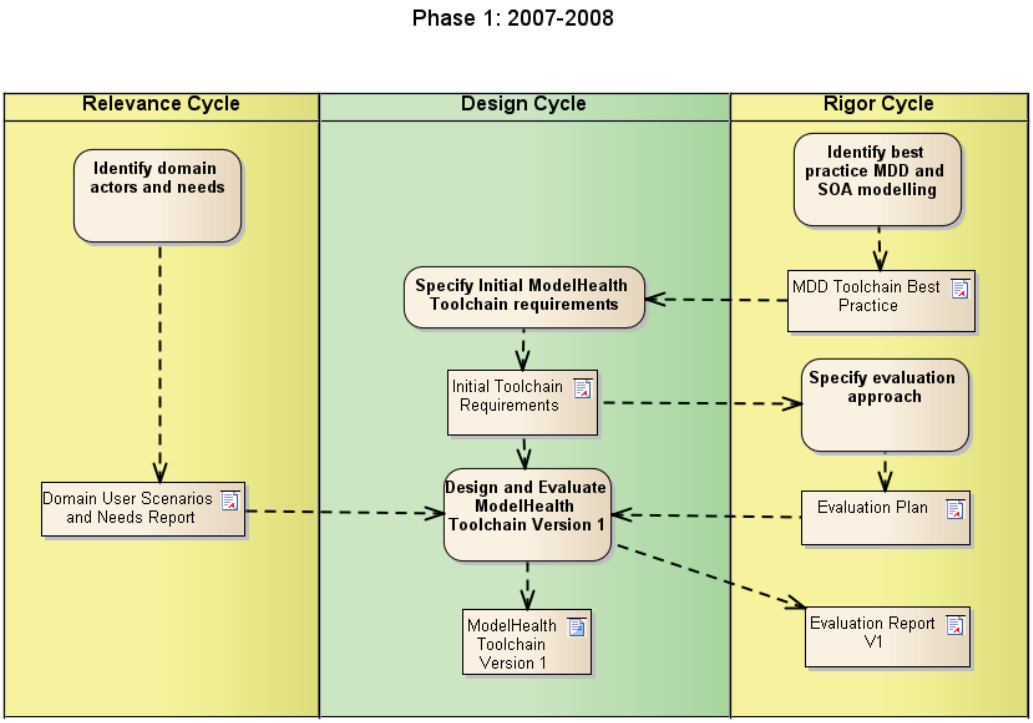

Figure 7: Diagram showing the first phase of the Design Science process. The work was carried out in 2007-2008

3.2.2 Phase 2: Design reusable services with new toolchain and develop pilot systems

The main objectives of the second phase were to refine the toolchain with a DSML based on evaluation from phase one, identify and design the reusable software services, and create an application based on these using the new version of the toolchain.

Table 5 The activities and results in Phase 2

Activity	Result	Comment
Identify reusable services	A set of generic domain services grouped in functional packages	The services are identified from a set of features derived from the use cases. Services are grouped into functional areas such as security, patient management and sensors. Details are presented in Paper 6.
Create reusable domain actor library	A UML profile with reusable UML actor modeling elements.	UML profiles can contain a library of reusable model elements (Selic 2007). These are made available to the developer when the UML profile is loaded in the modeling tool. Using best practice DSML design, two profiles are created. Details are reported in Paper 4 and 6
Refine and evaluate ModelHealth Toolchain V2	ModelHealth Toolchain V2 Evaluation report	The evaluation of the ModelHealth Toolchain after the first revision. Done with developers in the MPOWER project. Details are reported in Paper 3 and 10.
Model and implement Homecare SOA System	Specification of Reusable Models in Continuity of care Homecare Calendar and Message System	The defined list of service from the relevance cycle was used to design (model) and implement the services. Furthermore, a prioritized list of scenarios was realized with a SOA-based application. The application was developed by the development team in the MPOWER project and is described in Paper 7 and 9.
Design DSML for Continuity of care and SOA	ModelHealth DSML (design)	Use best practice DSML design and solutions to create a DSML for the target application domain. Process and results are presented in Paper 4.

An illustration of the relationships between the activities and results in the three cycles of Phase 2 is presented in Figure 8

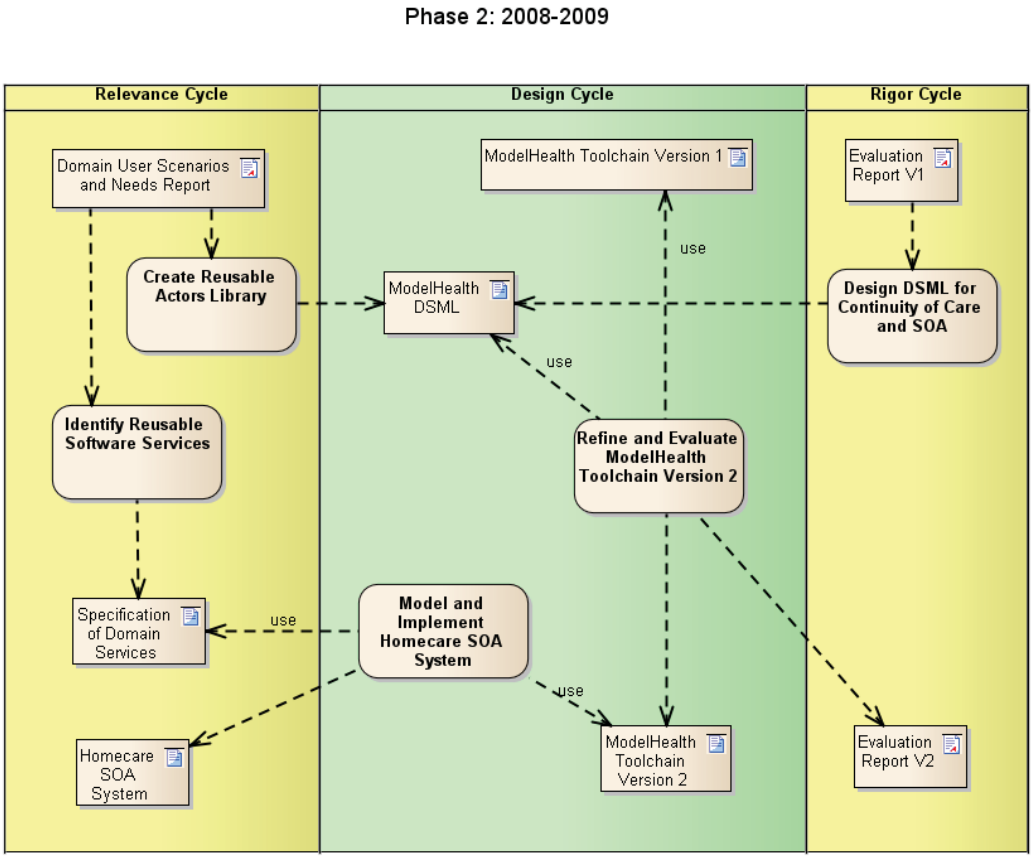

Figure 8: Diagram showing Phase 2 of the Design Science process. The work was carried out in 2008-2009

3.2.3 Phase 3: Evaluate pilot systems and final version of toolchain

The main objectives of the third and final phase were to refine and evaluate the third version of the toolchain, and evaluate the pilot systems.

Table 6 The activities and results in Phase 3

Activity	Result	Comment
Evaluate HomeCare SOA System	Homecare SOA Evaluation publication Open Source Domain Service Implementation	The implementation of the reusable software service designs was applied in a SOA application. The application realizes a number of the initial scenarios defined by the domain experts and was evaluated with in real life settings for one year. Details are found in Paper 7 and 9.
Refine and evaluate ModelHealth Toolchain V3	ModelHealth Toolchain V3 Evaluation Report Recommendations for Healthcare DSML Tool Support	Refinement of the ModelHealth toolchain, with a major update on the transformation module. Evaluation was done with students and professional developers. A set of recommendations is formulated from the evaluation results. Details are reported in Paper 10

An illustration of the relationships between the activities and results in the three cycles of phase 3 is presented in Figure 9

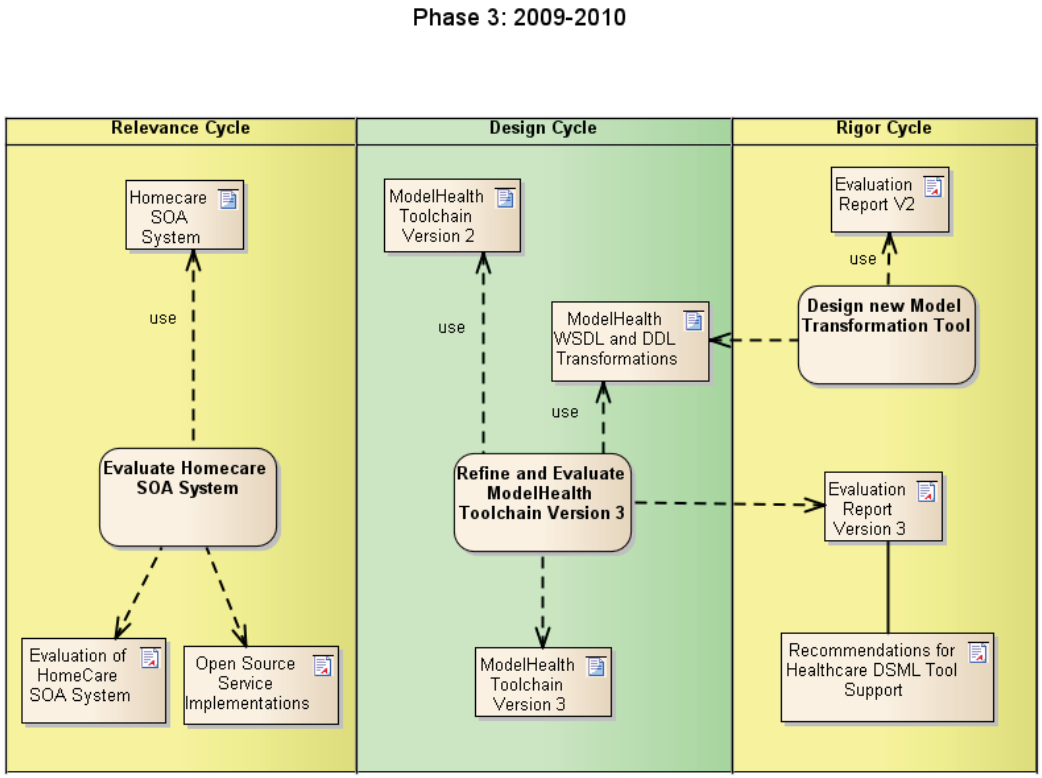

Figure 9: Diagram showing Phase 3 of the Design Science process. The work was carried out in 2009-2010

3.3 The ModelHealth Toolchain: design as a search process

This section presents the evolution of the ModelHealth toolchain as it was subject to refinements and assessments in three project phases. This section is a refinement of the toolchain development description from Paper 10. Figure 10 shows a summary of the toolchain version and evaluation process for each version.

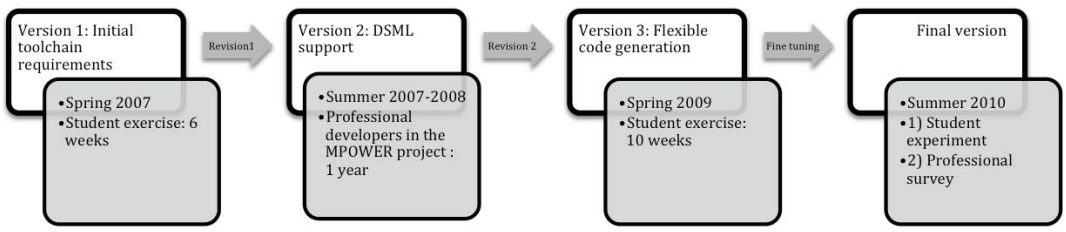

Figure 10: The ModelHealth Toolchain design cycle. Only a minor revision was required between Version 3 and the final version.

3.3.1 Phase 1: Designing the ModelHealth Toolchain

The work on the initial ModelHealth toolchain was carried out as a part of the MPOWER project that aimed at developing tools that enable rapid and robust development of smart home care systems. (The MPOWER Consoritum 2007). Developers from research and industry partners in the MPOWER project were involved in the requirements and design process. Table 7 shows the ten overall toolchain requirements that the MPOWER project defined for the model-driven development toolchain:

Table 7 The toolchain requirements

Req. #	Requirement	Details
TR1	The tools should be free of charge, preferably open source solutions	The tools should be possible to evaluate and use without any charge. Students and other small companies/organizations should see the toolchain as an affordable extension or alternative to existing development tools such as Eclipse IDE. Cost is an important factor in industry adoption (Staron 2006).
TR2	The tools should be easy to learn	The tools should be easy to learn and provide an intuitive interaction model to the developers. The ease of learning and use were also identified as important factors in the survey by Finnigan (Finnigan et al. 2000).
TR3	The tools should be easy to use	The tools should not require extensive setup or configuration to be used. Simple system designs should be simple to create, and complex system designs should be supported
TR4	The tools should run on standard computers	The tools should be possible to run satisfactory on a standard off-the shelf computer. Preferably on Linux, Mac OSX and Windows.
TR5	The tools should be extendable with UML profiles	The modeling tool should allow for domain specific language extension of the UML meta model using UML profiles. This requirement is inline with the recommendations for the "The Perfect Tool" that should allow for domain knowledge integration (MacDonald et al. 2005).
TR6	The tools should allow for addition and / or modification of transformation scripts	It should be possible to modify the model transformation scripts to fit the target domain and platform
TR7	The tools should generate documentation	The model created in the tools should be possible to export as documentation in different formats, minimum Rich Text Format (RTF).
TR8	The tools should be mature and in final	To reduce the risk for internal bugs in the tools, the tools should be in a mature state and have a significant user group.

	release	
TR9	The tools should support developers from project initiation to deployment and testing	All phases of a development process should be supported in a coherent way, from initial design/requirements specification through development, testing and to deployment. The tools should provide information to the users about the context and rationale for design decisions, preferably as a part of a traceability scheme.
TR10	The tools should support a top-down SOA based development approach	Though bottom-up and meet in the middle approaches are relevant, the target tool chain was designed to support top-down development. To incorporate domain concepts in the design it is recommended to do this in a top-down manner.

The research approach follows a design science paradigm where an artifact is designed and assessed in a search process (Hevner 2007; Hevner et al. 2004). Based on the ten initial toolchain requirements TR1-TR10, a tool and tool component investigation were conducted by a group of researchers from the MPOWER project group. The six project partners provided their preferred tools to create the target SOA artefacts and the results were summarized in a pros/cons matrix. As shown in Figure 11 the selection of tools for the first phase were:

- Sparx Enterprise Architect (EA) version 6.1 providing the Modeling Tool and Transformation Tool: EA is a mature tool, and expected to be easy to use and learn. It runs natively on Microsoft Windows, but can run on all other platforms using emulators. EA is not free of charge, but the price for a license was considered acceptable for both students and companies. In addition it did not require a powerful computer to run satisfactory. EA has a large user community that is active on the support forums. EA has a built-in tool and script for generating WSDL files, html documentation and RTF documentation.
- NetBeans version 5.5 with Derby DBMS and Glassfish v2.x (application server): It provides all required functionality in one installation and was considered easy to configure and maintain. A high-performance computer is recommended, but not required. NetBeans 5.5 was at the time awarded the best SOA IDE by Web Services Journal readers.[10]

The first design cycle was initiated with a student exercise employing Version 1 of the ModelHealth Toolchain as illustrated in the diagram in Figure 11. The EA WSDL script was modified by me to generate valid namespaces and correct basic types.

[10] Sys-Con website: http://tv.sys-con.com/node/171304

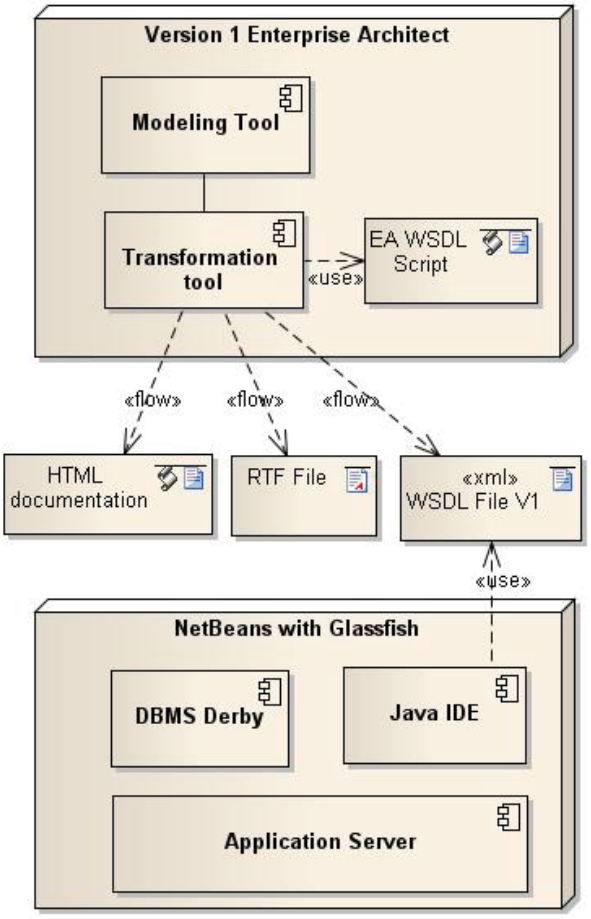

Figure 11 The first version of the ModelHealth Toolchain. Enterprise Architect provides the modeling tool and a transformation tool to generate HTML, RTF documents and WSDL files. NetBeans will use the WSDL file to generate a Web Service that can run on the Glassfish application server.

3.3.2 Phase 1: Evaluating ModelHealth Toolchain - Student exercise 1

As a part of the INF-3791 Telemedicine and e-health systems course (Spring 2008) at the University of Tromsø, Norway, a group of five students were introduced to the ModelHealth toolchain version 1. The assignment was to design a system for shared medication information. As input to the work they were given introductory lectures, and the lecture material was made available on the course's homepage. The course is international and the working language is English; hence all lectures, handout material and communication were in English.

The assignment was carried out as a 6-week project where a set of tasks structured according to the MDA approach (Miller,Mukerji 2003) should be completed and discussed in a weekly one-hour meeting. All students should participate in the meetings. During the weekly review meeting, the group presented and discussed their results and plans for the next period. Technical issues were discussed and solved.

- The results were documented in a UML model file, notes from the meeting and a final report.

- All students were present at the weekly meetings for presentation of their results and problems. The problems were in mainly two categories; UML notation and model structure.
- The theoretical introductory course in UML was not sufficient when it came to concrete modeling of system behavior and structure. The students referred to Internet tutorials and compared their design to example diagrams found online. In the context of MDD, model structure is of great importance to utilize model transformation and code generation. Reusing design elements across diagrams and model views is of utmost importance in order to maintain consistency and model traceability.
- The first version of the tool chain provided limited assistance for creating proper model structures. Reference material was given in separate documents and presentations as well as references to online material. Literature such as the SOA4HL7 guideline (Honey,Lund 2006) and the SOA book from Erl (Erl 2006) were found to be too comprehensive for this type of exercise and developer group, even though the main principles for service identification and design was presented in the lectures.

The students did not reach the implementation (coding) phase, but review of the system architecture and design found that the design model had sufficient detail and proper SOA structure to become a valid SOA system. The service candidate identification, interface specification, message design and service modeling were completed successfully and the design models would work fine individually as documentation and paper-based software specification using traditional programming tools. However, the lack of MDD experience made them create design models that did not have the necessary quality (completeness / correctness / consistency) (Mohagheghi et al. 2009a) to allow model transformation, code generation or even proper documentation generation.

The aspects of MDD that they found important were related to documentation and formalism. Modeling forced them to make a clear distinction between the target system and the environment (systems). Furthermore, a consistent and correct use of model element naming and the necessity of creating e.g. a complete information model were deemed useful. All properties of an element should be considered at design-time as the models are used for code generation. More advanced MDD functions such as UML profile use for improved code generation was not evaluated, but demonstrated in the final summary meeting. The verbal response from the students was positive.

In summary, the group of students having limited or medium software engineering experience successfully managed to design a SOA-based system for a given scenario specification. Due to the lack of UML/MDD training, their design final model was neither sufficiently detailed (complete) nor structured to be applicable for model transformation and code generation.

3.3.3 Phase 2 Revising ModelHealth Toolchain: Adding Domain Specific Modeling Elements in ModelHealth Toolchain

As a supplement to Sparx Enterprise Architect, it was considered important to support the developers in the modeling phase with a DSML for the relevant healthcare

subdomain, focusing on continuity of care. The DSML introduced in the tool supports the developer on two levels: model content (e.g., the Actor model elements shown in Figure 12) and project model structure shown in Figure 13.

- Project Model Structure: let the users load a preconfigured environment that defines required diagrams and model elements that and provide guidance on how it should be modeled and inspected (traceability view). The structure is part of a ModelHealth Project Template called "Service Project" that was installed in the modeling tool.
  - Traceability viewer: A configuration of the internal Enterprise Architect traceability tool to make it easy to visualize the relationships and dependencies between the core model elements. The diagrams and guidelines provided in the ModelHealth Service project template implicitly add traceability through the software design phases. If the user follows the guidelines and makes a complete design, using the traceability viewer allows the user to trace from initial scenario through use case diagrams, feature diagrams, information model and service designs, to code (WSDL). Details about this traceability scheme are described in Paper 9.
  - The project template:
    - creates a package structure and diagrams required for use case models, feature models, a information model and a complete service model. The diagrams have example model elements that clearly show how a service should be modeled. Notes in the diagrams provide short user guidelines.
    - Adds a new context menu for adding new services designs. A new "service design" creates a new package with the required service and message design diagrams and packages structure. This structure is specified by the IBM Software and Services UML Profile (Johnston 2005). The stereotyped elements were also loaded into a diagram-specific palette, showing only those elements that should be applied in the current diagram.
- Model content:
  - Loads stereotypes and tagged values for SOA service modeling from the IBM "Software Service UML Profile" (Johnston 2005). The profile elements are applied in the example models in the template.
  - As a part of the project template, loads a library of healthcare domain actors relevant for teamwork treatment and follow-up. These actors represent the main stakeholders and systems in the care domain, and are harmonized with international standards and known taxonomies on continuity of care and smart home environments as discussed in Paper 6.

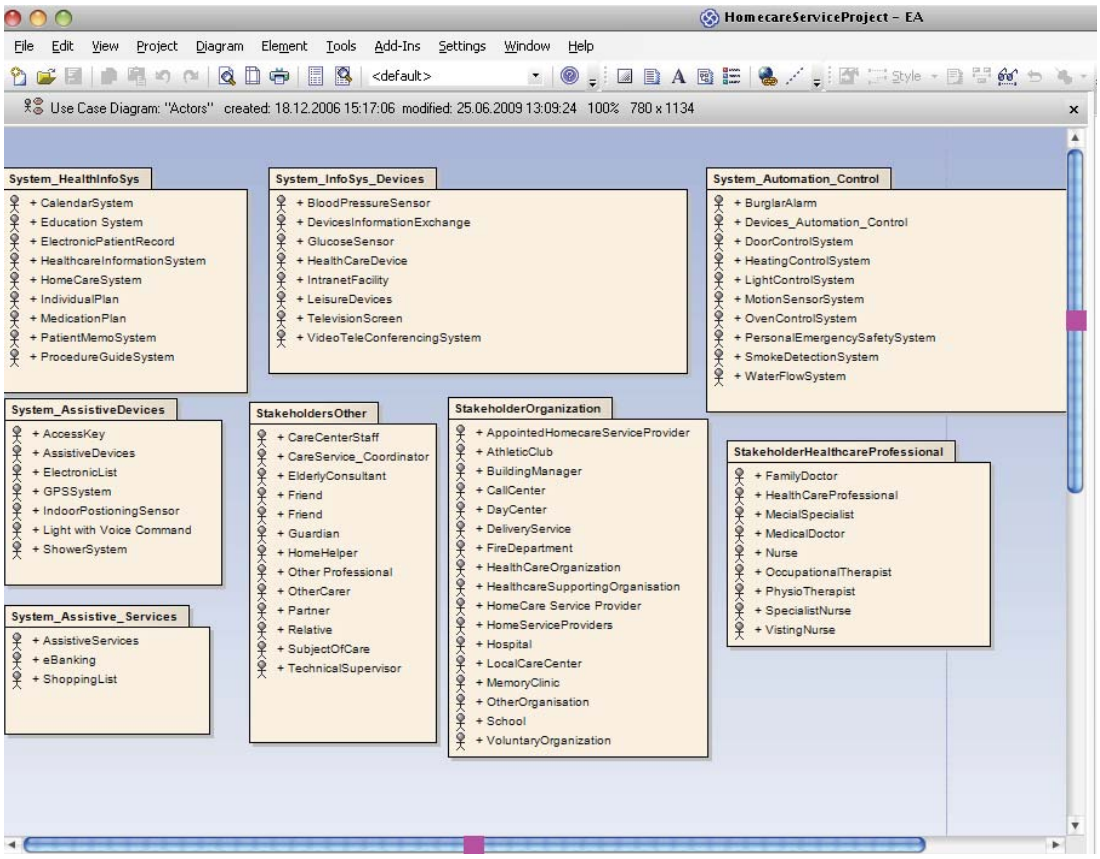

Figure 12 The initial project structure and content. The actors library contains eight sub-packages to make it easier to navigate and find the appropriate actor element.

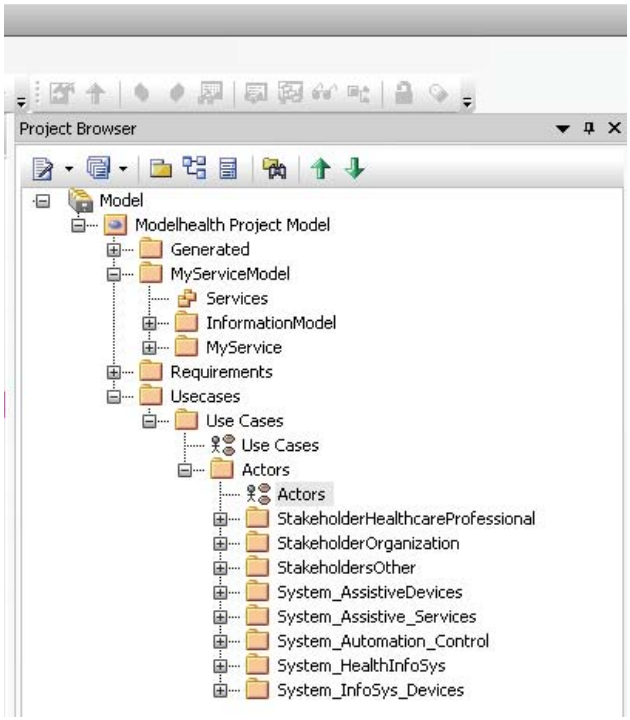

Figure 13 The initial project template structure. In each package there is a sample diagaram that provide instructions on how to carry out the modelling

Version 2 of the ModelHealth toolchain in shown in Figure 14 illustrating how the modeling tool makes use of the project template and software service UML profile.

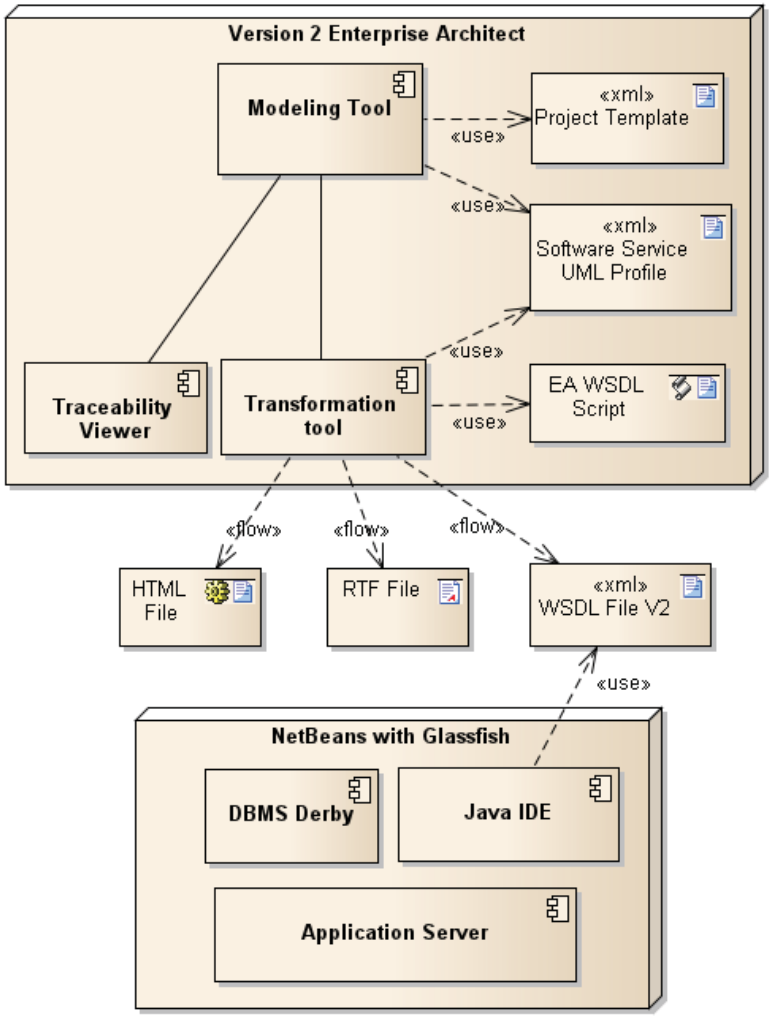

Figure 14 Version 2 of the ModelHealth Toolchain. The Project template and the UML profile (with actors library) is used by the Modeling Tool. The Traceability viewer provided in EA allows for simple coverage and orphan analyses.

When the developer starts the modeling tool (Enterprise Architect), the project browser will list a structure that has packages for each development phase and diagrams that must be completed in order to get a complete design model (see Figure 13.) The domain actor library provides a list of UML Actor elements named and described according to the domain standards. The diagram in Figure 12 shows the actor library elements.

The actors library is mainly used for use case and feature modelling, associating activities and features (requirements) with actors. When the developer has completed the usecase and feature modeling, the information model and service modeling is done using the Software Service profile structures as shown in Figure 15. Service interface operations and input/output message types can be easily defined using the tailored element palette as shown in the left toolbox.

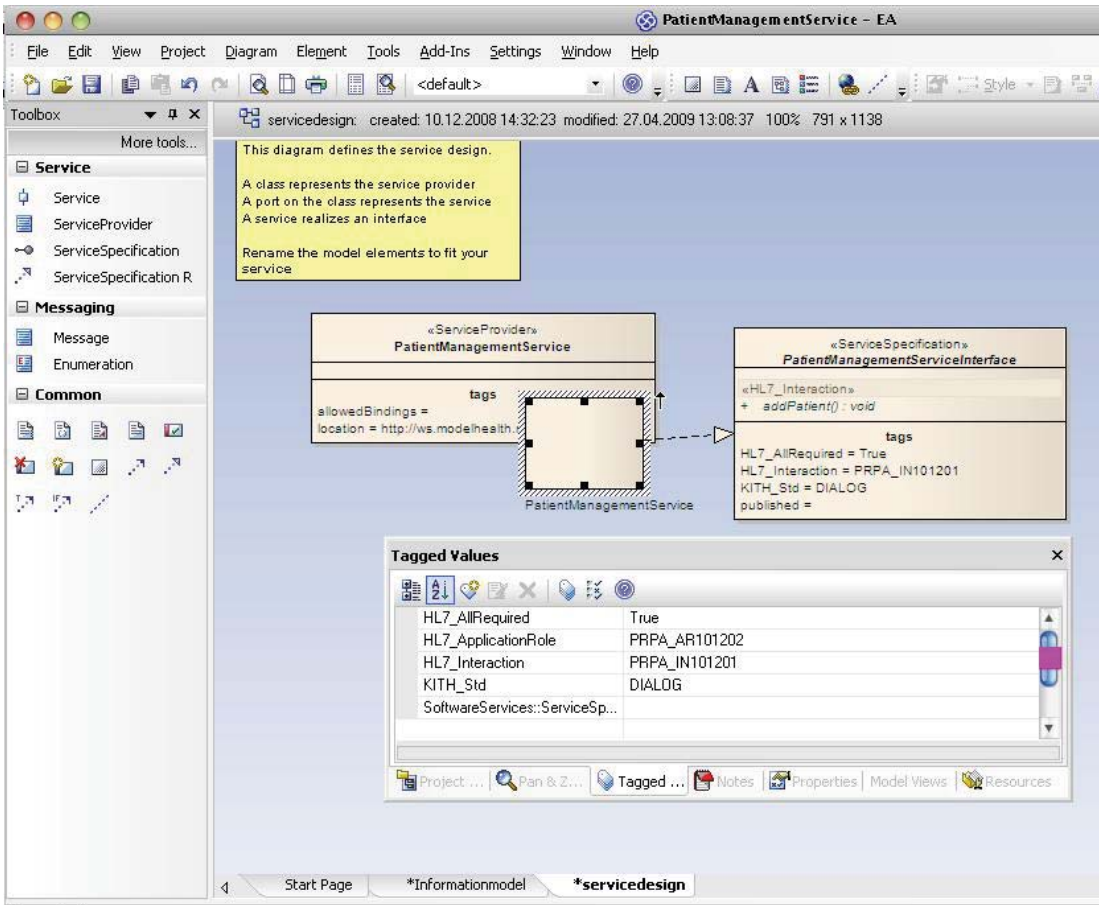

Figure 15: Example of service design showing the PatientManagementService and its interface

As described in Paper 12, the coverage and orphan analysis traceability services are provided by a simple relationship matrix as shown in Figure 16. Coverage analysis gives an overview of which modelling elements (e.g. feature) that are related and which are not related to other elements in a proceeding development phase (e.g. realizing elements such as service). Orphan analysis gives an overview of model elements that are not related to other elements, especially elements in the previous development phase (e.g. services that are not connected to a feature). For each feature (rows) that is supported by a service (columns) there is an upward arrow. For details about the tracelink, the developer can double-click the arrow to inspect the details about the tracelink and navigate to the link ends, e.g., view the details about a feature for a feature-service tracelink.

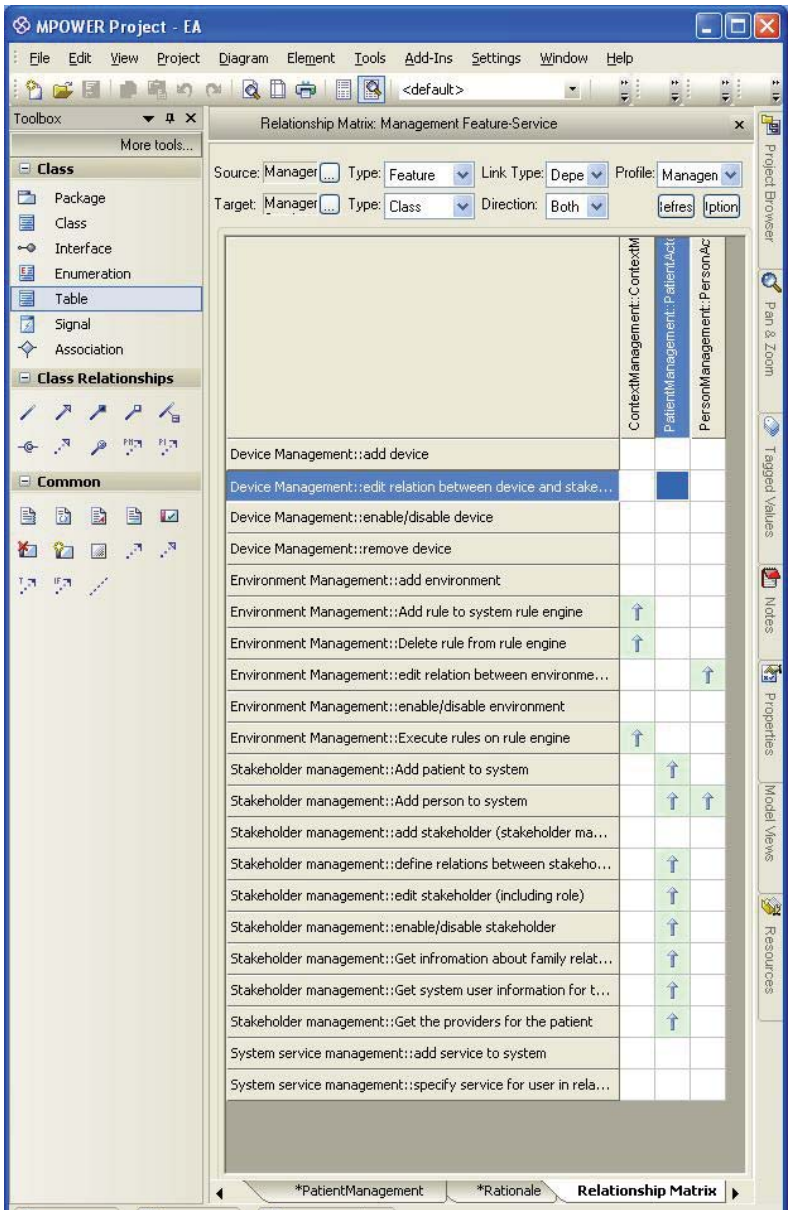

Figure 16: Example of a relationship matrix showing traceability coverage of features to services

When the service designs are completed, code and documentation can be generated using actions available in the modeling tool menu system.

3.3.4 Phase 2: Evaluating the ModelHealth Toolchain in the MPOWER project developer study

As part of the MPOWER project (The MPOWER Consoritum 2007), a comprehensive SOA design model was created using version 2 of the ModelHealth toolchain. The design model included:

- 18 user scenarios, each with three to four sub-processes were described. These were then subject for analysis by system architects using UML use case modeling.

- 113 UML use cases were described from the scenarios, separated in 10 groups.
- 82 actors (stakeholders and system actors) were identified and modeled.
- A total of 145 features were derived from the use cases and structured into the logical groups.
- 25 software service designs were created to realize the features

The evaluation of the MPOWER project developers study was published at the European Conference on Model Driven Architecture in 2008 (Paper 3). A questionnaire was sent to each developer to which all developers responded. The questionnaire was structured according to the technology acceptance model addressing the easy of use, usefulness and compatibility with daily work processes (Davis et al. 1989; Venkatesh,Davis 2000). The results show that the model-driven toolchain supports the development, although some errors in the code generation process can make the developers spend much time on debugging code that should be flawless. Inherent aspects of model-driven development such as traceability and generation of system documentation were found useful.

The evaluation results show that: *"the respondents indicate that MDSD tools must be perceived useful and should be easy to use. Tool performance does not have a direct effect on MDSD use, although business analysis, traceability and code generation were found useful. It is especially important that MDSD tools are stable and provide complete and correct artefacts"*. The main shortcoming of the toolchain was the inflexibility of the built-in EA transformation component: *"Using Model-Driven development improves my job performance and productivity, only if everything works well with the transformation of models... Otherwise you can find yourself spending too much time trying to make things work (and doing the required changes manually). If this is the case then using Model-Driven development takes too much time from my normal duties."* (Walderhaug et al. 2008a)

3.3.5 Phase 3: Revising Toolchain: adding generic model-to-text transformation

After the first student evaluation and the MPOWER evaluation it was found necessary to replace the built-in transformation in the chosen UML modeling tool with a stand-alone open solution. This solution was chosen to be Eclipse 3.3/3.4 (Java IDE and modeling tools) with MOFScript plugin version 1.3.2. This is a generic model-to-text engine that allows the developers to traverse the model and output any kind of text. A script for MOFScript that generates a WSDL was developed and incorporated into the Eclipse Project along with the necessary DSML support files.

To export the UML design models from Sparx Enterprise Architect to Eclipse, it was necessary to develop a transformation stylesheet in XSLT. This required a significant effort and resulted in a 1600 LoC stylesheet that transforms the EA XMI representation into a XMI representation that Eclipse supports. The developed stylesheet solution is a major extension of the solution presented by Kahn et al. in (Khan 2008).

In addition, the internal SQL Script in EA was modified to support Derby databases, allowing for easy database development (from the information model) and web

service testing on real data. The third and final version of the ModelHealth toolchain is shown in Figure 17.

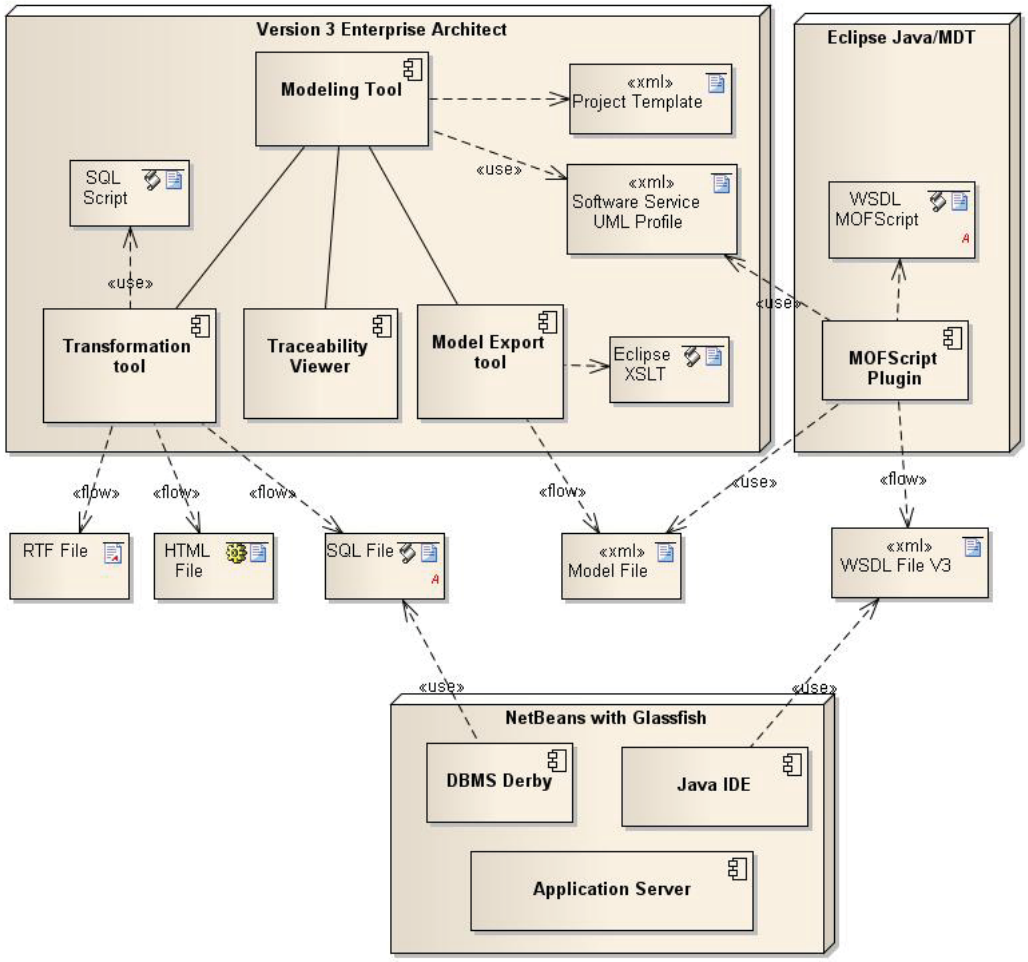

Figure 17 Third version of the ModelHealth Toolchain. Eclipse and MOFScript was introduced to generate a error-free WSDL file. Support for Derby DB SQL was added

3.3.6 Phase 3: Evaluating the ModelHealth Toolchain - Student exercise 2

Version 3 of the ModelHealth Toolchain was evaluated in a 10-week student project at the University in Tromsø. The development process was followed up closely by using an agile development process with weekly standup-meetings (scrums). Ten students in the Telemedicine and eHealth course (autumn 2009) were given an assignment to develop a SOA based Shared Medication List system using the ModelHealth toolchain version 2.

As preparation for the project, the students were given lectures on MDD in general and the toolchain specifically. The ModelHealth Service Development Guideline (D1) and the instruction videos (V1) were available on the course's homepage. Paper 10 provides a detailed description of the evaluation process.

The students were divided into two groups based on a screening process of development skills and experience. The assignment and process was described in

detail and divided into three main phases. In addition, an optional phase was added in case there where time. The phases were:

1) System design, modeling and transformation
2) Database creation, Web Service implementation and deployment
3) Application development
4) Change request and reimplementation of web services

A separate room was reserved for the project, with two high-performance workstations set up for each group. Two whiteboards in the room were used to track the design and development tasks using multi-colored stickers.

The results were documented in the EA UML model file, notes from the Scrum Product Backlog and a final report. A brief evaluation of the process and results for each phase is shown in Table 8.

Table 8 Summary of student exercise 2 results

Project development phase	**Task**	**Comment**	**Domain knowledge, MDD utilities and information needs**
System Design and modelling	*Use case, information model, service designs*	*Design quality OK. Some orphan elements (not correctly deleted) had to be fixed by supervisor*	*The students used the scrum meetings to discuss attribute details, associations (relationships) and naming conventions.*
Model transformation	*Generate documentation, database schema and WSDL files*	*All generation OK*	*Students needed assistance when generating the WSDL from Eclipse.*
Database creation, Web Service Implementation and deployment	*Create Derby database, generate web service skeletons, implement web services, deploy and test.*	*Database creation OK*	*Students commented that they had to spend time on populating the database with test data. Would like to have this generated from an object model.*
Application development and deployment	*Create a SOA desktop application in NetBeans*	*Partly OK. GUI ok, controller logic required debugging*	*Students not familiar with Hibernate and database connections. Some assistance on Java programming resolved issues.*
Change request	*Update information model and service design. Transform new artefacts*	*Artefacts retransformed, database and web service updated.*	*Students were really happy to see how fast changes were implemented. Main problem was that the database had to be repopulated with test data.*

Only minor bug fixes was done to the ModelHealth V3 toolchain after the second student exercise.

3.4 The ModelHealth Service Design process

The process of designing domain services with the ModelHealth toolchain is based on the design guidelines in (Erl 2006; Honey,Lund 2006) .The process is a top-down approach that normally starts with a specification of the target environment, its requirements and concerns. Fig. shows a diagram of the main process divided into three core phases. These phases correspond to the CIM, PIM and PSM levels of OMG's MDA approach (see section 2.3.1). Each step in the process in described in Table 9.

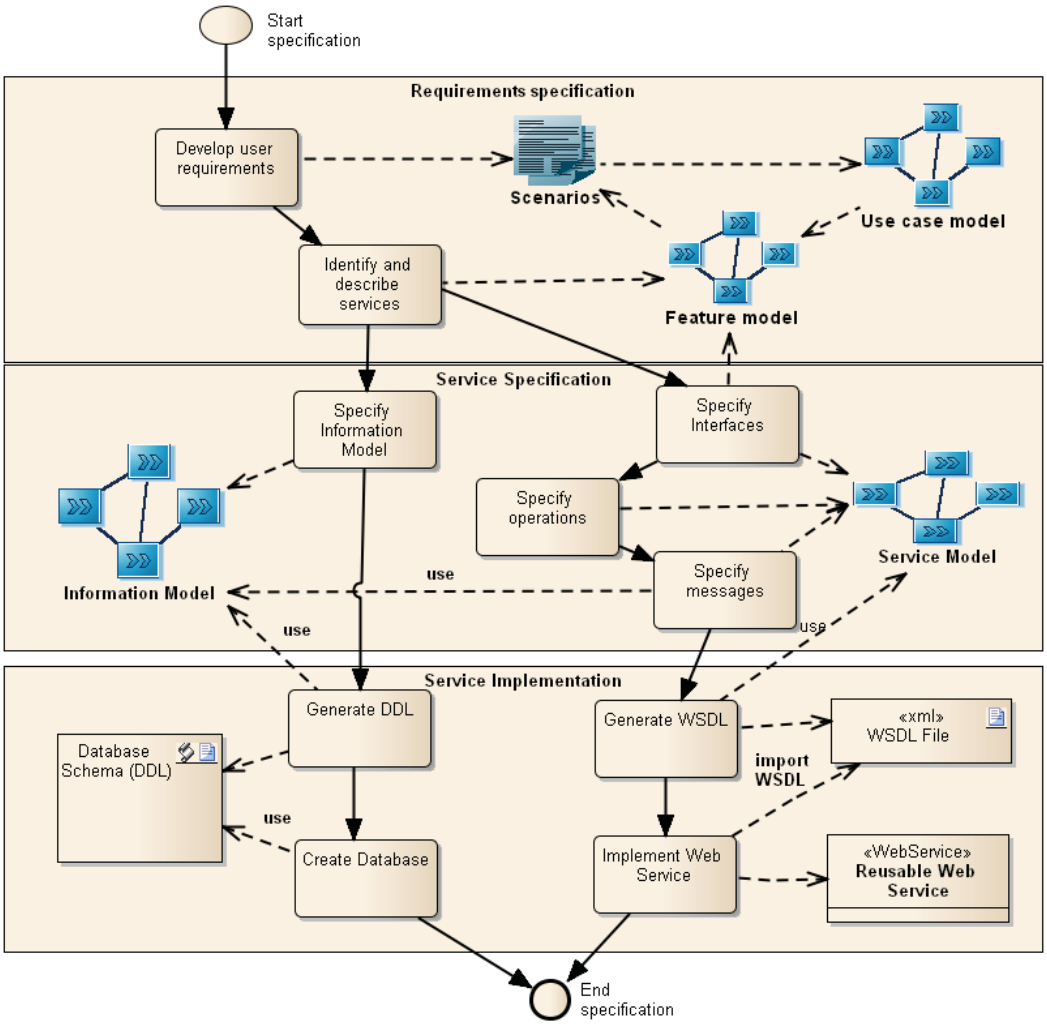

Figure 18 Service development process

The software service development process is divided into three main phases: Requirements specification, service design and service implementation. In each phase, a set of diagrams are modelled and tracelinks are created.

Table 9 Summary of main steps and artefacts in the service modelling process

Phase	Activity	Artefact (diagram)	Comments
Requirements Specification	Develop user requirements	Scenario descriptions, use case models and system feature models	The scenarios should be included in the use case properties (supported by UML modelling tool). Features must be derived from use cases using <<trace>> dependency link.
	Identify service candidates	Service classes and interfaces (ports)	Group features and identify service candidates. Separate read and write interfaces according to best practice SOA design in (Erl 2006) and SOA4HL7 in (Honey,Lund 2006). Each service must realize one or more features that must be modelled as a Service class – Feature <<trace>> dependency.
Service Specification	Specify information model	Information model	Create a domain information model that allows for specification of relevant request and response messages (documents) for the interface operations. Use case and feature model element should be used as input.
	Specify interface operations	<<Service Specification>>	Use feature descriptions to specify operations on the interfaces. Add operations to the interface as needed.
	Specify service messages	<<Message>> diagram	Use the <<Message>> element from the tool palette to create a message diagram. Add properties to messages using information model elements as types in addition to UML primitive types. Iterate this process until all interface operation parameters are set using the message model elements
Service Implementation	Generate WSDL and web service	WSDL file and Web service skeleton	Use the ModelHealth WSDL transformation to generate WSDL files for each service (multiple service designs are supported in one model). Use your preferred development IDE to generate Web Services skeletons for implementation.
	Generate DDL and database	Database description file (SQL) and	Generate DDL from the information model. The DDL file can be used by database management tools to create

		database instantiation	database instances.
	Generate service documentation	RTF document and html project (with links)	Use the documentation generator to create complete service documentation in RTF documents and HTML projects. The HTML project provides traceability navigation.

3.5 Evaluation methods and data collection

This PhD project follows what Oates refers to as a "design and creation" research strategy. This strategy *"…could offer a construct, model, method or instantiation as a contribution to knowledge. Often the research outputs are a combination of these*"(Oates 2006). The ModelHealth Toolchain and the reusable software services designs must be designed and evaluated using sound evaluation methods.

Selecting the correct evaluation method is essential for the quality of the evaluation results. In (Easterbrook et al. 2008) Easterbrook et al review a set of empirical methods and the process of selecting the appropriate method for different types of research questions, theoretical stances and practical considerations (e.g., access to subjects and resources). For each evaluation method, different data collection methods can be used. In (Sim,Lethbridge 2008) , Sim and Lethbridge present a taxonomy for field study data collection techniques. The taxonomy divides the techniques into three main categories: *direct, indirect* and *independent* techniques. Each technique has advantages and disadvantages, and can be used individually or in combination to "*allow for a more accurate picture of the studied phenomena"* (Sim,Lethbridge 2008) (page 30).

Details about the evaluation methods and data collection techniques are not provided in detail here as the books "*Guide to Advanced Empirical Software Engineering*" by Shull et al., (Shull et al. 2007) and "*Researching Information Systems and Computing*" by Oates (Oates 2006), give excellent explanations of these.

In the following subsections, the evaluation method and data collection techniques are briefly presented for the two core artefacts: the ModelHealth Toolchain and the set of reusable software services (see section 1.3). Detailed references to the appropriate literature are given.

3.5.1 ModelHealth Toolchain evaluation

As described by Easterbrook in (Easterbrook et al. 2008) it is often useful to apply more than one technique to investigate a research question. A mixed-method approach was used for evaluation the ModelHealth toolchain because of the exploratory nature of the research questions and uncertainty of the access to resources, such as developers to participate in experiments. The two methods mixed were (ibid pages 294-303):

- Action research: the toolchain was developed in three iterations involving two groups of masters students in informatics, and 16 professional developers in the MPOWER project. Collaborating with the students and developers, the problem of utilizing the benefits of domain specific model driven development was investigated by designing and implementing software services with the ModelHealth toolchain. Data from the students projects was collected using focus groups as part of a Scrum process. To capture the data from the professional developers, a questionnaire based on the "Technology Acceptance Model" (Davis et al. 1989; Venkatesh,Davis 2000) was used.
- Controlled experiment: the final version of the ModelHealth toolchain was used in a controlled experiment with university students in the master's program in informatics. Data was collected using observation notes on "*Inspection Data Form*" (Seaman 2008), semi-structured interviews (both direct techniques), and development result analysis (independent technique). In addition, the students were instructed to "think-aloud" during the experiment (Van Someren et al. 1994). Interview data were transcribed and analyzed using the "*constant comparison method*" (Miles,Huberman 1994).
- Survey: a survey with professional developer of healthcare information systems was conducted after a demonstration of the final version of the toolchain. Data was collected from a total 25 participants representing the target user group for the toolchain, using a questionnaire addressing domain specific MDD and SOA for healthcare.

Paper 10 describes the methods and materials for the toolchain evaluation in detail.

3.5.2 Software service evaluation methods

The process for evaluating the software services was to follow a rigorous design approach and apply a reference implementation of the services in the development of two pilot systems. Figure 19 from Paper 7 shows the main steps in defining and evaluation the services.

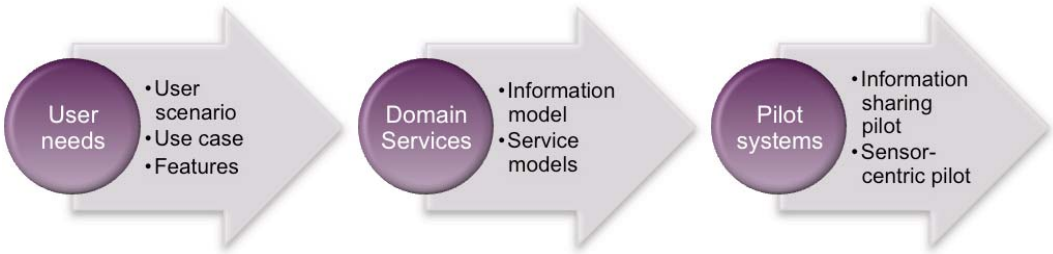

Figure 19 The main steps in defining and evaluating the reusable software services

With reference to Figure 19 and Figure 20 (also in Papers 3 and 6):

- The "user needs" were developed from user workshops, expert interviews, literature study and user questionnaires. The results were documented as "user scenario specifications".
- Doing use case modelling produced a use case model (including features) and actors model.

- From the features and use cases, a set of services were identified and designed in a service model.
- Using model transformation, WSDLs were generated and used to create reusable web services
- Building upon the reusable web services, two applications were developed and deployed.
- Finally the applications were evaluated in realistic environments.

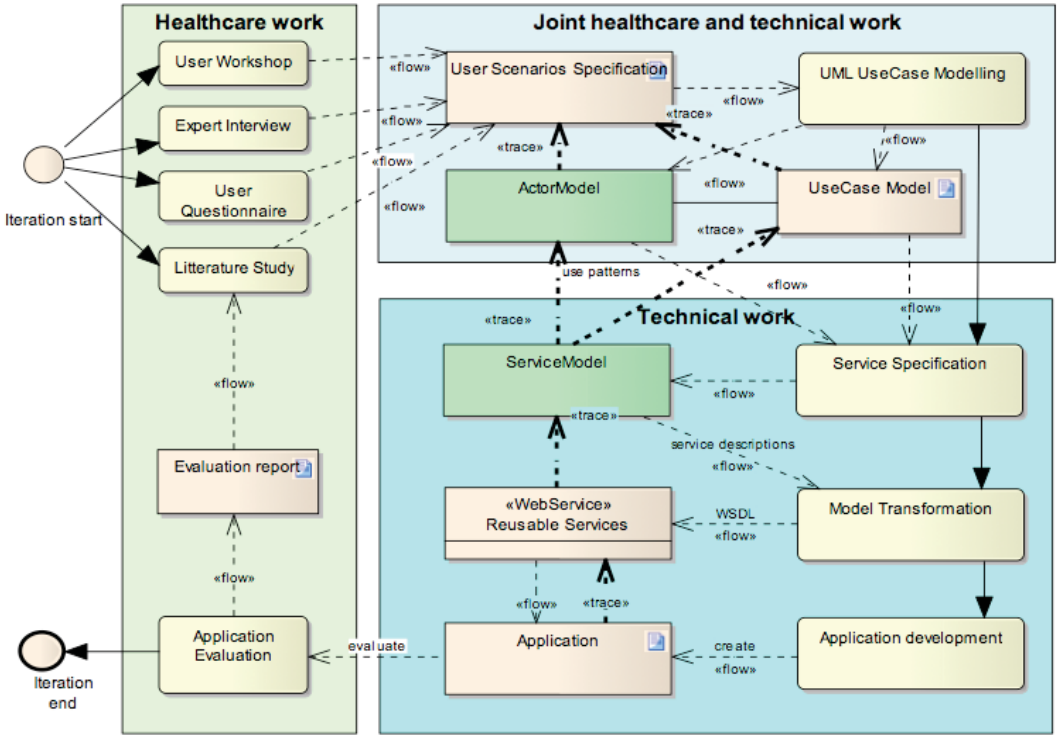

Figure 20: Detailed view of the design and development process for reusable software services and pilot systems

To evaluate the software services, a case-study approach was applied. Two pilot systems realizing a prioritized set of user scenario were developed using the reusable software services. Practical considerations made it impossible to do long-term evaluation on both pilot systems since they were deployed to two different geographical locations; Norway and Poland. The system deployed to Norway was evaluated using training sessions and follow-up interviews per phone by the vising nurse.

To collect evaluation data about the system development and technical functionality a combination of direct and independent techniques were applied. After the system development a questionnaire was submitted to the developers addressing the use of SOA in systems development, in terms of the MPOWER project. The results were analyzed using descriptive statistics. To complement this "picture", experience from technical difficulties with installation and operation was logged, e.g., deployment difficulties and network issues.

3.6 My Role in the MPOWER Project

The work described in was carried out within the frames of the MPOWER project. Many of the activities outlined in section 3.5.2 were carried out in collaboration with other researchers and programmers in the project. The list below clarifies my role in these activities. With reference to Figure 20:

- I was involved in the literature study and questionnaire design to the domain experts.
- I contributed to the scenario descriptions (activity and problem) and led the work on use case modelling
- I made the actor model as described in Paper 6.
- I was strongly involved in service identification process and led the modelling of services in Enterprise Architect.
- I was responsible and the creator of the information model, with contributions from Ericsson Nikola Tesla in Zagreb, Croatia.
- I was the creator of the model transformation mechanisms, with valuable input from Dr Erlend Stav, SINTEF.
- I was the lead programmer for the web service handling all person, patient, provider, user and relationships information.
- As the technical manager I was the scrum master for application development. TSB Solutions and Dimension-Informatica in Spain, Ericsson Nikola Tesla in Croatia, University of Cyprus and AIT in Austria were the main application development partners.
- I was responsible for deployment, testing and maintenance of the Norwegian Pilot system for one year. I also attended evaluation sessions in Norway together with Torhild Holthe and Ingrid Haug-Olsen from the Norwegian Center for Dementia Research.

Chapter 4 Results

The results are organized according to the three research questions. For each research question, the *key findings* are presented with references to the publications from which they are synthesized. Each question is discussed in separate subsections: 4.1 addresses how a MDD toolchain can assist developers in creating reusable services, 4.2 addresses how domain standards and knowledge should be imported into a MDD framework, and finally subsection 4.3 addresses which reusable services that are relevant for care and management of elderly living in their homes. A summary of the key findings is shown in Table 10

Table 10 Relationship between the findings, the papers and the research questions

#	Finding	Addressed in paper(s)	Research Question(s)
F1	Continuity of care standard concepts relevant for service design can be modeled as UML Profiles	P1, P2, P4, P6, P10	R2
F2	Ease of use and correct code generation is important for the usefulness of MDD tools	P3, P10	R1
F3	Traceability services are considered an important utility in healthcare software development and can be provided using basic UML dependencies or using more sophisticated trace models	P3, P5, P8, P10	R1
F4	The toolchain should provide project structure and process assistance.	P3, P5, P10	R1
F5	Presentation of domain information in the design tools should be flexible, consistent and easy to use	P10	R2
F6	The modeling tool should provide design model verification and validation	P4, P10	R1
F7	A relatively small number of reusable software services cover a large part of the ICT support needs for independent living	P6, P9	R3
F8	Simple service-based applications have the potential to support older people at home, particularly older people with memory problems who need support in structuring the day and keeping an overview of the daily activities and appointments	P7, P9	R3

For each research question discussed in the following, a UML class diagram illustrates the relationships between the concepts. *The dashed arrow from finding to research question e.g. F2 to R1 means that finding F2 contributes to answering the research question R1. The dashed arrow from e.g. paper P3 to finding F2 means paper 3 provides rationale for the finding F2*. These relationships are shown for all papers (P1-P10), findings (F1-F8) and research questions (R1-R2).

4.1 R1: How can a model-driven development toolchain with domain support assist the development of reusable domain software services?

Using MDD to develop software services involves the creation of UML models and model-to-text transformations. The challenge addressed in this doctoral project is how to design a toolchain that software developers with limited MDD experience can utilize to improve the process of developing reusable software services. The target toolchain is called the ModelHealth Toolchain.

The Design Cycle described in 3.1 explains how the ModelHealth Toolchain artefact was created and assessed in three cycles. An essential part of the ModelHealth Toolchain is the approach to software services design where the developer identifies key system features from which services can be derived and designed. To complete the service design model, a set of diagrams must be created according to a required structure. The final design model is then used as input to code and documentation generation. The ModelHealth service development process is described in section 3.4.

From the toolchain assessments carried out in the design cycle, four findings were specified as shown in Figure 21. In the following, each finding is discussed in terms of its underlying investigations.

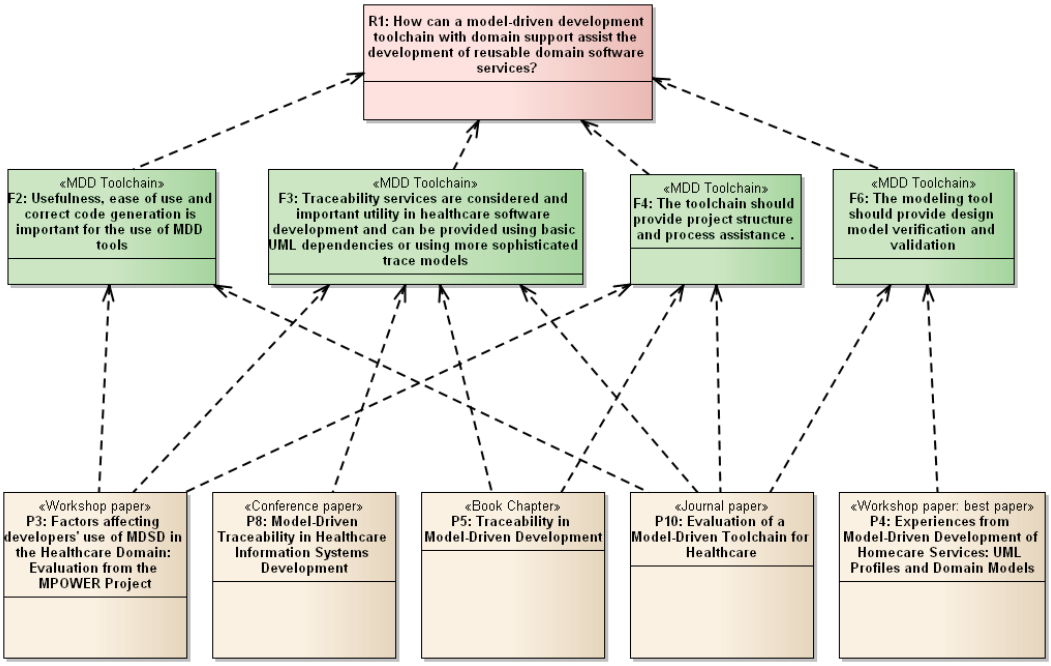

Figure 21: Overview of the relationships between research question 1, the relevant findings and publications reporting them

4.1.1 F2: Usefulness, ease of use and correct code generation is important for the use of MDD tools

In the MPOWER project, 16 developers used the ModelHealth Toolchain Version 2 for designing reusable software services to support continuity of care. The design and development process was carried out during a period of 12 months, resulting in 25 software services. Paper 3 reports from a survey done on this developer group using an online questionnaire. The results from the survey identify usefulness, ease of use and "correct code generation" as the essential factors for using MDD.

One of the developers explicitly states that:

"Using Model-Driven development improves my job performance and productivity, only if everything works well with the transformation of models... Otherwise you can find yourself spending too much time trying to make things work (and doing the required changes manually). If this is the case then using Model-Driven development takes too much time from my normal duties."

The refinement of the toolchain for version 3 resolved the problem of generating incorrect code. In Paper 10, the results from the student experiment shows that no problems were encountered during transformation and that generated artefacts such as code and documentation were correct. The evaluation results also revealed that the participants' design model quality was sufficiently good to generate useful WSDL files, databases and documentation of these. In the follow-up interviews, three main utilities of the toolchain approach were identified: improved system overview, code and documentation transformation, and traceability of artefacts.

Paper 10 reports from an "expert opinion" workshop where professional developers of healthcare information systems were asked about their attitudes towards MDD in

healthcare. The developers were given a demonstration of the ModelHealth Toolchain (version 3) and responded to a paper-based survey and gave oral feedback to the workshop facilitator. The responses showed a clear indication that they believed that MDD would be useful as it could improve the quality of the work process and results. The majority would like to learn more about MDD, as the current use of core model-driven development techniques was low.

4.1.2 F3: Traceability services is considered important an important utility in healthcare software development and can be provided using basic UML dependencies or with more sophisticated trace models

Traceability has been identified as a core utility of MDD in the literature. However, a unified traceability scheme is missing, and cross-tool traceability is hard to achieve. Paper 5 provides a comprehensive specification of a traceability solution that would allow for domain specialization and traceability information sharing. The proposed solution supports the core traceability services:

1) **Trace inspection**: the purpose of trace inspection is to allow the trace user to inspect trace information to get a better understanding of (parts of) the system and its development, both during development and maintenance. Trace inspection functionality should include the ability to visualize, navigate, and query traces.
2) **Coverage analysis:** through coverage analysis, the trace user can determine the degree to which some artifacts of the system are followed up by other artifacts in the system.
3) **Orphan analysis**: Orphan analysis is used to find artifacts that are orphaned with respect to some specified trace relations. The analysis should be able to find single orphaned artifacts, but also isolated groups of artifacts with trace relations of the specified types only internally in the group.
4) **Change impact analysis**: One use of trace information is to determine the impact a change to an artifact will have on other artifacts. The results of a change impact analysis can be used to estimate the cost, resources and time required to perform the change, or even to determine if the change can be allowed or realized at all.

Paper 5 describes a complete *Traceability System* (page 12) specifying how traces between artefacts can be created according to a *Trace Model* and shared through a *Traceability Repository*. The paper presents a complete system example demonstrating the proposed solution.

In Paper 8, the importance of traceability is discussed in terms of software validations recommendations from the Food and Drug Administration (FDA) (U.S. Department Of Health and Human Services et al. 2002). FDA strongly recommends applying a traceability scheme for managing requirements fulfillment, component dependencies and testing. Applying one simple and one extended *Trace Model,* the paper demonstrates how the aforementioned traceability services can be realized in the ModelHealth Toolchain for web services development.

In the MPOWER developer study (Paper 3), the student experiment and the professional developer study (both Paper 10), traceability services were outlined as

important utilities improving development process, system understanding and documentation. The professional developers report that traceability tools are used by a few projects, and mostly for requirements to tests, but that MDD could improve the use of traceability services.

In the ModelHealth service development process, some tracelinks had to be specified manually, such as service to feature realization. This manual step was neither considered time-consuming nor problematic, and the value of having a complete trace model largely outweighed the efforts required to maintain it (Paper 10).

The investigations done in the project employed only a simple traceability scheme. Paper 8 shows how the solution described in Paper 5 can be used for creating valuable information for use with change impact analysis in particular. The traceability analysis results can be displayed in several ways, and Paper 5 illustrates one way based on UML Profiles and stereotype icons as shown in Figure 22. Here the amount of effort required is illustrated using colored dashed dependency links – black is "no impact, green is "low impact" and red is "high impact". It can be easily detected that the "define relationships between stakeholders" feature is the most difficult or complex feature of the PatientActorControl software service.

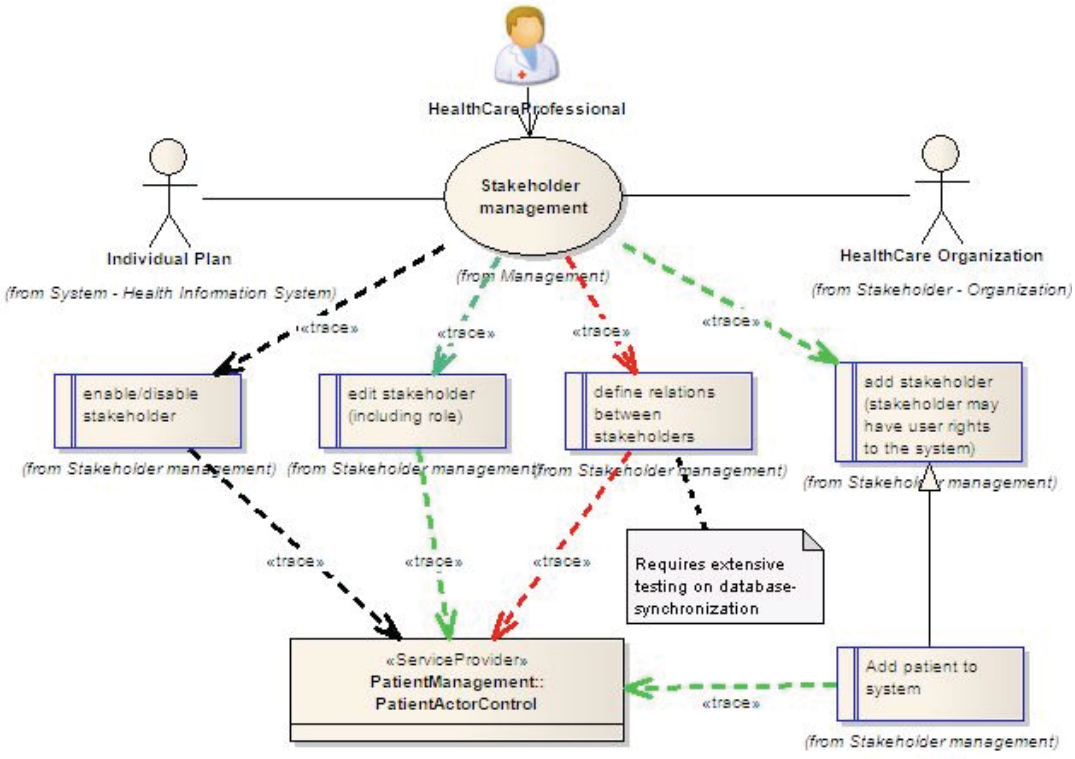

Figure 22 Example of stereotyped change impact analysis results from Paper 5

4.1.3 F4: The toolchain should provide project structure and process assistance

A fundamental utility of MDD is the ability to generate development artefacts such as code or scripts. As discussed in the previous section, it is of utmost importance that the generated code is 100% correct so that the developer will not need to debug

generated code. When designing the ModelHealth Toolchain it was experienced that developers could introduce unwanted model elements and create incomplete designs. Hence, the generated code would become incomplete or incorrect.

The refinement from Version 1 to Version 2 introduced project templates, model palettes and diagram examples (see section 3.3). The 16 service developers in the MPOWER project used Version 2 for designing 25 software services. The evaluation of 12 months design work found that much coordination was required for a distributed service design project (Paper 3, Table 7). Having a common structure and process for service design allowed for the complete design of 25 services derived from a common actor model, use case model, feature model and information model.

The MPOWER developers identified traceability as a useful utility (Paper 3, Table 7), improving the development process and understanding of the system. To fully benefit from the traceability solution described in Paper 5, a common project structure and process must be implemented. The quality of the traceability services strongly depends on the completeness and correctness of the information in the *Traceability Repository*.

In Paper 10, the design model qualities are evaluated in terms of correctness, completeness comprehensibility, confinement and changeability. The students participating in the experiment had limited experience in using MDD, and used the ModelHealth toolchain Version 3 to create service design models having sufficient quality for successful code and documentation generation. This is in large contrast to the students with comparable experience using Version 1 of the toolchain that were unable to create models that could be used for code generation.

The experience from developing the toolchain and the final student experiment show that novice MDD users can make effective use of a MDD toolchain with sufficient project structure and process assistance. This finding is also proposed as a toolchain requirement in Paper 10.

4.1.4 F6: The modelling tool should provide design model verification and validation

Providing project structure and process assistance will help the developers create models that are applicable for code generation. However, the developer must still select the correct model elements, give them a proper name, associate the element to other elements and keep the model file clean of orphan elements.

To validate models with respect to quality, will in most cases require human effort. As stated in the SEQUAL framework: "*In the area of semantic quality, general automated tools are difficult to develop for the simple reason that the domain and audience are beyond automatic manipulation*" (Krogstie 2012). One way to address this problem is training and adapting the modelling language to the domain. In terms of UML, this can be done using the UML profile mechanism.

In Paper 4, the development of two UML profiles for SOA and HomeCare software services is presented. The profiles contain domain specific constraints that should be met in the service designs. These constraints, specified in OCL, should be validated at modelling-time. As an example, one of the constraints requires that a patient

<<SubjectOfCare>> element in a diagram should have at least one association to a provided carer <<HealthCareProfessional>> element. A MDD toolchain should provide functionality for easily validating such constraints. Figure 23 shows the proposed UML profile for Homecare.

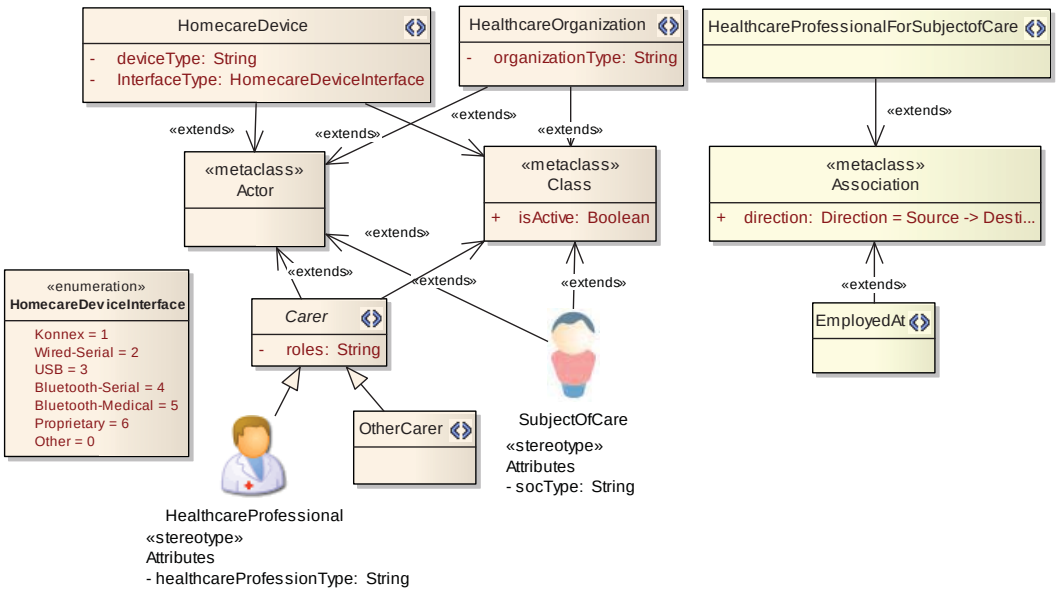

Figure 23 Proposed UML profile for Homecare presented in Paper 4

In addition to validation model constraints, syntactic model correctness can be verified at modelling time. Simple modelling errors such as having duplicate elements with the same name in the same package, properties without a specified type, and orphan elements in the model file, should trigger a warning from the tool. Such errors were encountered during the student exercises in the toolchain design process described in section 3.3.6. In this case, the supervisor found the model errors during model review in the scrum meetings.

In the student experiment reported in Paper 10, one of the student groups designed an incorrect information model in the sense that it was not applicable for database structure generation (DDL file). In the experiment, the round-trip time from model to code was very short, allowing for quick fixes in the model without introducing ripple effects in the coding. In a larger project this banal error in the design would have caused unnecessary work for many people.

Related to validating model correctness is the ability to utilize traceability services at modelling time. These should be readily accessible from the tool, allowing the designer to navigate in the model, monitor coverage (e.g. requirements), and detect orphans (e.g. features not realized or services not connected to features). In the aforementioned student experiment, it was found that trace navigation using a html browser was useful for maintaining a system overview.

This finding is one of the toolchain requirements reported in Paper 10, and is further supported by discussions in Paper 6.

4.2 R2 How can relevant domain standards and knowledge be incorporated into a model-driven development toolchain, and what aid can they provide in the design and development process?

There is a common agreement that standardization is an important criterion for creating interoperable systems. The concept of continuity of care as defined in the introduction should be supported by interoperable information systems allowing involved stakeholders to access and update information relevant for the care provision.

The vast majority of Standards Development Organizations (SDOs) provide their standards mainly as documents. Software developers must read and understand these documents and develop code that adheres to the recommended structures and best practices. In many cases, design flaws will be revealed during integration testing.

The ModelHealth toolchain address these challenges by incorporating relevant standards and best practices into developers' tool suite. Evaluation of how this solution worked identified two key findings discussed in the following. Figure 24 shows the relationships between research question 2, the two findings and the papers.

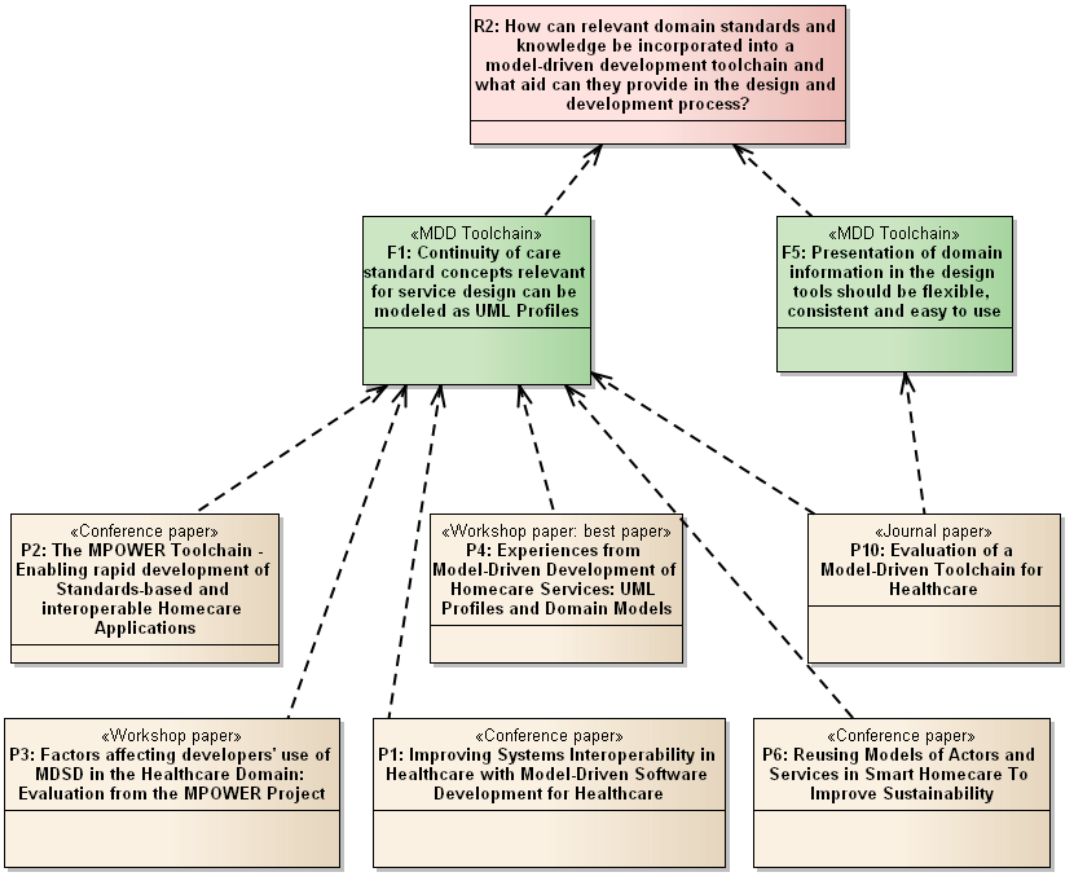

Figure 24: Overview of the relationships between research question 2, the relevant findings and publications reporting them

4.2.1 F1: Continuity of care standard concepts relevant for service design can be modeled as UML profiles

The overall concept of how domain specific knowledge could be incorporated into a MDD toolchain was explained in Paper 1 providing a "CarePlan" example profile with model transformation. In Paper 2, the process and first experiences from using the toolchain with HL7 messages in the MPOWER project are reported. From these initial investigations and ideas, the UML profiles were further developed and explained in Paper 4. Figure 25 shows the overall process followed for developing the UML profiles. This process is a key result from Paper 4 as such processes are poorly explained in the literature (Selic 2007).

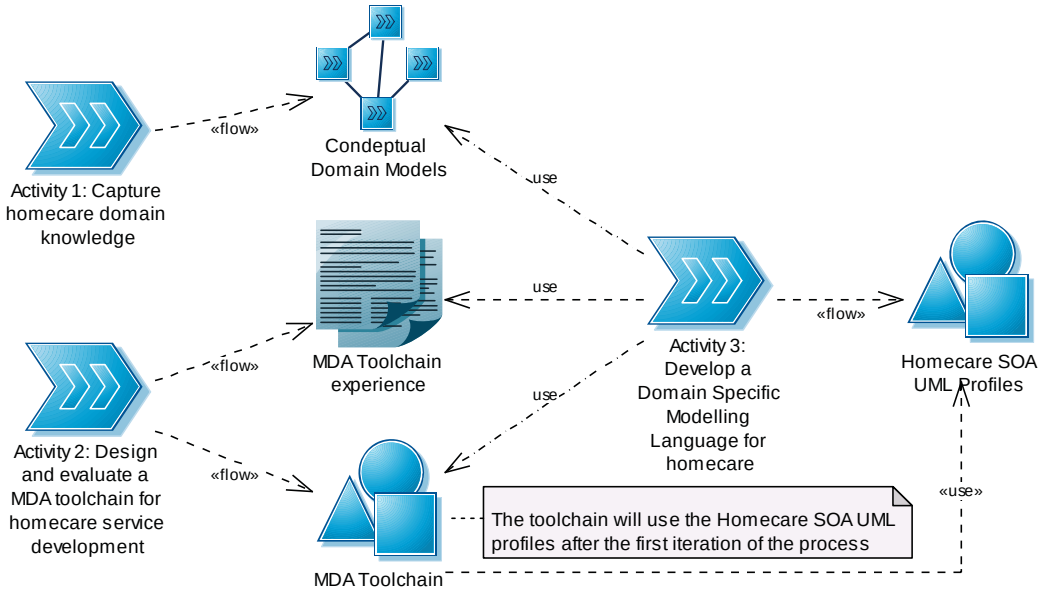

Figure 25 Process of creating UML profiles for homecare and SOA domains from Paper 4.

The results are two UML profiles:

- Homecare UML Profile: a profile classified as Virtual Metamodel Extensions (Staron 2005). This implies that this profile is mainly used to increase the expressiveness of the modelling language when designing systems for homecare. A "virtual metamodel extension, restrictive" stereotype adds a domain specific icon such as a picture of a nurse to the modelling element, together with a well known domain specific label such as HealthcareProfessional.
- SOA HomeCare UML Profile: a profile including elements from the "Code generation, restrictive" category [ibid]. These stereotypes can improve code generation by providing domain information so that code generation scripts can create high-quality code.

The use of the profiles is discussed in Paper 4, but only reduced versions of the profiles were subject for a developer evaluation. In the evaluations presented in Paper 3 and Paper 10, the core elements from the SOA HomeCare UML profile were

available together with an extensive "library" of reusable model elements. The reusable elements are presented in Paper 6 and include 78 actor elements grouped into 8 packages. The actor elements are based on the CONTSYS standard, specializing e.g., the HealthcareProfessional actor into a homecare actor labeled VisitingNurse.

The main evaluation of the UML Profile with the reusable elements is presented in Paper 10. The unanimous response from the experiment participants is that it was very useful for building a design model with the "correct" domain concepts. All the five student groups reused the required actors model elements from the library, and only one group found it necessary to create two new actor elements.

To summarize, a UML Profile providing key concepts from the CONTSYS standard (CEN TC251 2006) was successfully developed and evaluation showed that it was suitable for designing a relevant domain information system. The process of creating a domain profile for healthcare can be applied other standards and domains.

4.2.2 F5: Presentation of domain information in the design tools should be flexible, consistent and easy to use

Observations and interviews reported in Paper 10 comprise the main justifications for this finding. In the student experiment, integrating domain specific information into the models was in some cases perceived cumbersome. The domain actor library containing 78 elements was hard to navigate, and the element names, though based on the CONTSYS (CEN TC251 2006) standard, were not always adequate for making a precise decision on use.

As a result, Paper 10 proposes two detailed recommendations for incorporating domain information in a modelling tool:

- *The DSL should provide a natural structure of reusable domain specific elements*. The elements should be named according to best practice from the domain and it should be easy to get more information about the element and its use. A detailed definition with examples of use is advisable along with the possibility for keyword search.
- *The DSL should provide a mechanism for adapting the element naming and structure to different information standards*. As there is a plethora of information standard in healthcare, often covering the same domain area or concepts, the DSL should allow for updates and switching of these, without having to rebuild the tools or refactor existing models.

4.3 R3 Which reusable software services are relevant in the care and management of elderly living in their homes?

An important part of the doctoral project was the implementation and evaluation of applications based on reusable software domain services. Figure 31 on page 39 shows how the relevant need from domain actors (relevance cycle) were incorporated into the design cycle as domain scenarios and needs, actor selection and definitions. Based on this, a prioritized set of scenarios and services were implemented in the SOA-based homecare applications, referred to as Proof-of-Concept-Applications (POCAs).

The findings supporting Research Question 3 are based on evaluations done of the POCA development process and real-life use by elderly and their carers. Figure 26 shows the relationships between the research question, findings and publication.

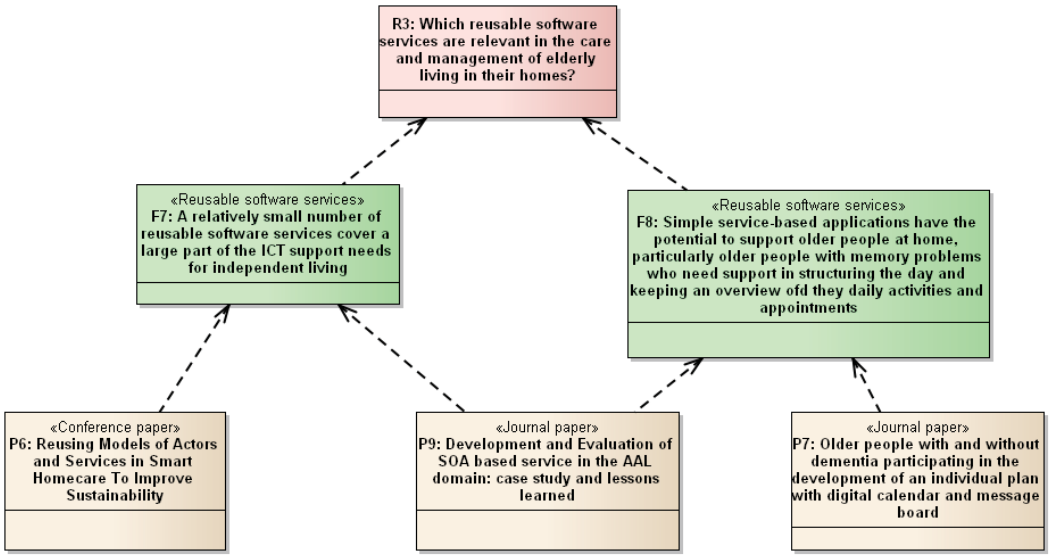

Figure 26: Overview of the relationships between research question 3, the relevant findings and publications reporting them

4.3.1 F7: A relatively small number of reusable software services cover a large part of the ICT support needs for independent living

Paper 6 describes the process on how 18 problem and activity scenarios developed by elderly and their families, caregivers and domain experts. (each 2-4 pages long). These scenarios were analyzed, resulting in 145 domain system features. A feature represents a specific system support functionality and must be related to one or more use-cases. A use case is based on one or more activity scenarios that involve at least one actor. Figure 22 shows an example of how the service for patient management is related (traced) to its origin.

As described in Paper 9, applying the recommended best practice for service candidate identification, a total of 25 service candidates were designed iteratively in the ModelHealth Toolchain.

The first preliminary version of the services is presented in Paper 6 whereas Paper 9 gives a detailed description of the final version. A condensed summary of the 25 services described Paper 9 is provided in the following.

4.3.1.1 Medical and social information services

The medical and social middleware services design represents the cornerstone of the Subject of Care individual plan support provided. Among the provided services are the Calendar and Medication management features, which are central in this context. The following services are offered:

- *Medication management* provides functions for managing and retrieving medication information for a subject of care. HL7 supported.
- *Calendar management* provides functions for scheduling all kinds of social and medical activities for subjects of care, caregivers, family, and friends. HL7 supported.
- *Message board* provides functions allowing caregivers or family members to exchange messages that could contain any patient's related information that needs to be shared. HL7 supported.
- *Reminder* provides set of operations for creating and managing various types of reminders for upcoming medical and social activities and events. HL7 supported.
- *Patient management* provides information about the patients through a common and standardized interface, and enables the developers to add, update and delete stakeholders from the system. In addition, the interface allows for querying for relationships between stakeholders such as patient- provider relationships or patient-relative relationships. HL7 not supported.

4.3.1.2 Communication services

The communication services support different kinds of communication between users and systems, including alarm handling, sending messages and notifications, and calls with voice and video. The following component and services are offered:

- *Alarming* service is designed to manage alarms in the system, and provides operations to trigger new alarms, accepting and deactivating alarms, as well as querying for current alarms and their status.
- *Notification* mechanism: To receive the notification message from the system environment the application has to subscribe to the notification mechanism using the notification service.
- *External notification* implements web-methods to interface external service for sending emails and SMS messages, by connecting to email servers and a public HTTP2SMS service.
- *Voice-/video communication* services provide the possibility to call other users (audio live stream) and watch them (IP-camera live stream). The SIP-based service includes methods for managing incoming, outgoing and active calls, and for managing accounts and contacts.

4.3.1.3 Sensor framework services

The sensor services provide functionality for configuring (add, remove, adjust) devices and retrieving sensor information. The following component and services are offered:

- *Frame Sensor Adapter (FSA) framework service* provides unified access to sensors and actuators that use different communication channels and different data formats.
- *Door control* provides a service for accessing and operating a door lock.
- *Door control management* provides functionality for manage access to different areas of a house.

- *Camera Access* is used for controlling and providing access to a camera stream via a HTTP network protocol.
- *Device manager* is used to register several types of devices that are to be installed in a system, including device types and protocols.

4.3.1.4 Interoperability services

Interoperability services are providing an interface for external systems. This is important, as medical and social relevant data have to be transferred to legacy systems, etc. The following services are offered (see also Fig. 8):

- The *Export to Google Health* service: is providing all medical and social relevant data in a standardized data record format.
- The *Medication Plan Synchronizer* Service: offers the functionality to synchronize data of the internal MPOWER system with data records of any legacy system (hospital information system or nursing sys- tem) using CCD or CCR.
- The *iCal and Google Calendar Export* service: provides the possibility to export and synchronize the calendar information from the internal Calendar Management service with external calendar systems.
- With the *Calendar Synchronizer* a subscription to the iCal format is possible. With such a subscription applications can get the latest updates of events and reminders by downloading and parsing the iCal file.
- The *UDDI Service Registry* service provides a platform independent way of describing and discovering Web services and Web service providers.

4.3.1.5 Security services

The security middleware is orthogonal to the other services in the way that it is implicit a part of each service, ensuring a satisfying security level of any combination of services in system. Authorization is based on a Role-Based Access Control (RBAC) scheme; a set of permissions is associated with each defined role, and users get permissions indirectly through the roles they are assigned. The following services are offered:

- *Authorization service*: determines what operations and which data an authenticated user can access, allowing access to resources only to legitimate, authorized users.
- *Authentication service*: verifies a user's credentials and allows access to the system only to users with valid credentials.
- *Token management service*: is used by the authentication and authorization services to manage the login sessions.
- *Role management service*: enables the Administrator to manage the roles of the system. The Administrator may add/delete roles, assign users to roles, get the role information, and get the user's assigned role.
- *Access management service*: manages the permissions and access profiles associated with the access control system.

- *User management service*: enables the Administrator to manage the users of the system. The Administrator may: add/delete users, update the user's roles, and get the user information.

The services presented are all provided open source through the FREE MPOWER project (MPOWER Consortium 2008a) and are designed using the ModelHealth Toolchain.

4.3.1.6 Using SOA in development

Paper 9 reports from the development of the two SOA based pilot systems. The 16 developers involved in systems development filled in a questionnaire addressing the perceived ease of use, usefulness, compatibility, and future use intentions. In addition a set of questions about the claimed benefits of SOA were included.

The developers were positive to SOA development and planned to use SOA in future development projects. A core advantage of the SOA architecture in the case of pilot systems development was that the system through rapid development comparatively easy could be tailored into similar systems, concerning the GUI or the interface in general to the user. Likewise, in situations where only parts of the functionality provided by an application is needed, it is easy to reuse selected parts (e.g., services or service compositions) to produce a tailor made application with specific functionality.

4.3.2 F8: Simple service-based applications have the potential to support older people at home, particularly older people with memory problems how need support in structuring the day an keeping an overview of the daily activities and appointments.

As presented in section 3.5.2, two pilot systems (Norway and Poland) were implemented using the reusable software services as foundation. Figure 27 shows a screenshot from the Norwegian pilot system annotated with references to the underlying reusable web services. Paper 9 describes how the pilot systems' functionality is closely linked to the underlying services' functionality.

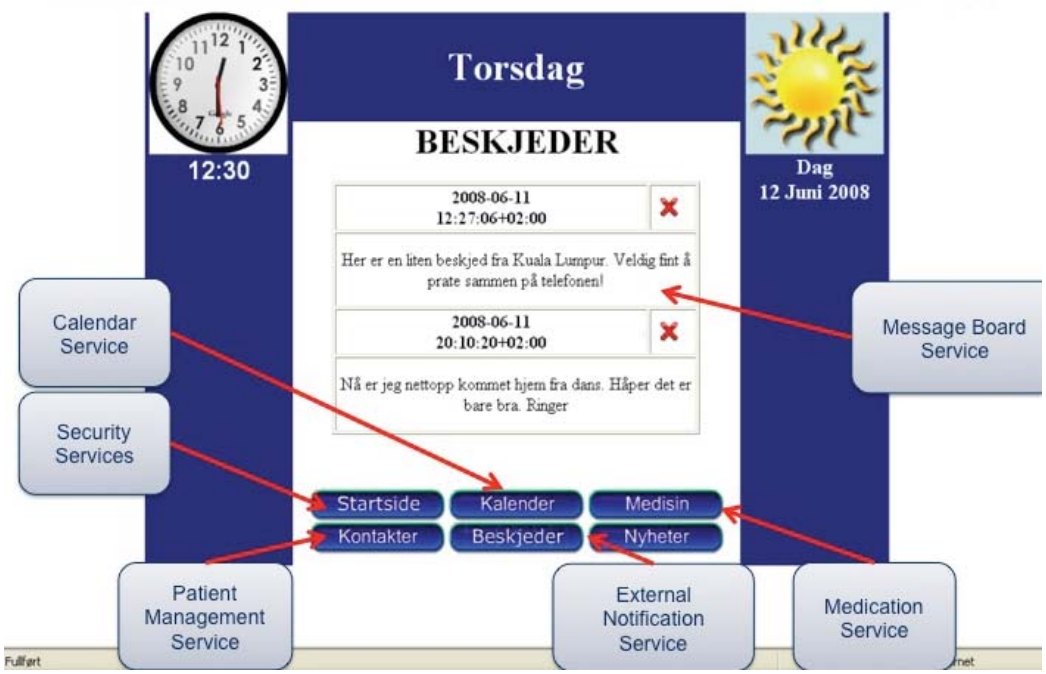

Figure 27 An annotated screenshot from the Norwegian pilot system. The underlying services are linked to elements in the user interface

The pilot systems were deployed to real sites and evaluated together with local care providers and the patients' family. Due to both organizational and technical challenges, the Polish pilot system was only evaluated from a human-computer interaction viewpoint that is outside the scope of this project.

Paper 7 reports from the trials done with the Norwegian pilot system (NPOCA). Seven older people, aged 65–92 tested the pilot system at home. Five of these had some memory problems. Figure 28 summarizes the evaluation results.

Table 2 Overview of use and utility of POCA during the second iteration December 2008. Case histories of use and utility in third iteration

Respondent	Time for testing NPOCA	Uses NPOCA	Utility of POCA	Comments
NOR02	10 months	Yes, every day	Yes, needed the screen to verify which day it was	'The calendar page is the most useful for me. I compare it to my paper calendar.'
NOR02Carer	10 months	Yes, once or twice a week	Yes, it gives us something to talk about when I call her	'I put all messages to mother on the calendar page.'
NOR05	7 months	Not the last 5 months	No utility because nobody adds appointments	Might have been beneficial if family members had added appointments
NOR05Carer	7 months	No	Have not used it	Uses e-mail instead
NOR06	3.5 months	Yes, every day	Very useful. Looks at the screen every morning in order to know what is to happen that day	'This is my pal – I look at the screen every morning.'
NOR06Carer	3.5 months	Yes, a couple of times a week	Very useful	'I think mother has become more aware of her surroundings since having had the screen.'
Domiciliary services	10 months	Occasionally	Not for me, but I assume user and carer benefit most from it	Difficult to use at patient's home, because the keyboard and mouse are hidden and I have to use them on the floor or on my lap = tricky!
Project worker	10 months	Yes, twice a week	Very useful for those in need of help with remembering things	Family carer should be in charge of adding messages

Figure 28 Table 2 from Paper 7 showing the evaluation summary of the Norwegian pilot system

Finding 8 is based on the knowledge about the underlying technical structure of the Norwegian pilot system and the conclusions made in Paper 7 saying that the system: *"was an innovative approach that definitely enabled older people with memory problems to live independently at home."*

Chapter 5 Discussion

The previous section described the eight findings from developing the ModelHealth Toolchain, designing the reusable software services, and implementing and evaluating SOA-based pilot systems supporting continuity of care. In Figure 29, the overall problem model presented in Figure 2 is refined with the results presented in section Chapter 4.

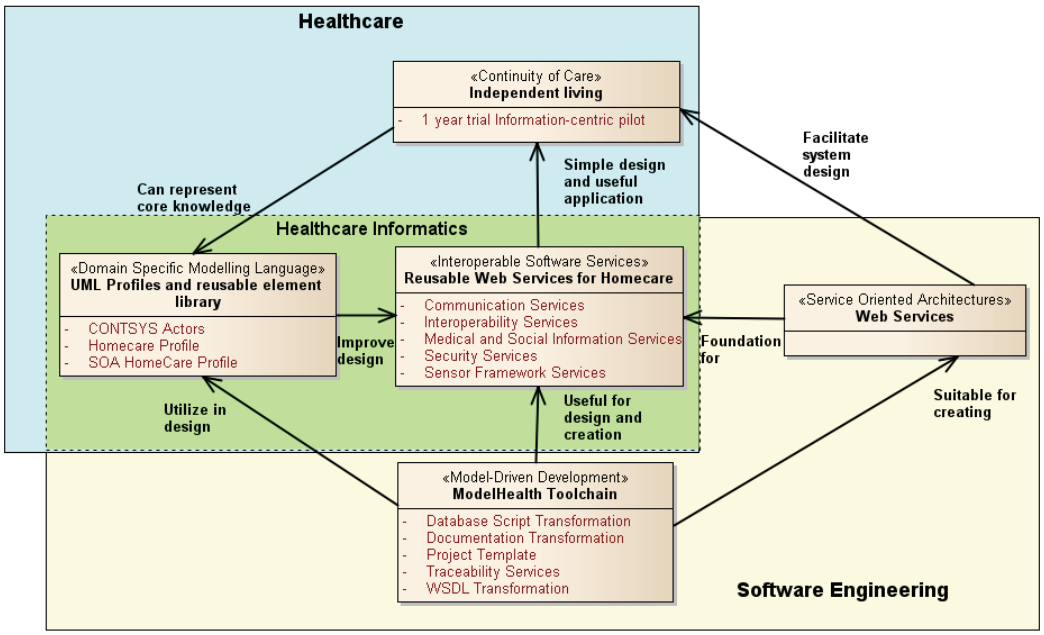

Figure 29 The problem model from the introduction instantiated with the results from the project investigations

The original concepts from Figure 2 are used as <<stereotypes>> on the result concepts, and design artefacts such as the actor library, UML profiles, the script and the service categories are shown as attributes on their owner result concepts:

- **Independent living**: building upon continuity of care, an information-centric pilot system was evaluated in a 1-year trial in Norway. The knowledge about the needs and standards in the domain was incorporated into the UML Profiles and reusable element library.
- **Reusable Web Services for Homecare:** 25 services supporting continuity of care and independent living were successfully applied in creating an independent living application – the Norwegian pilot system trial. The services are provided as open source in the FREE MPOWER project at SourceForge (MPOWER Consortium 2008a).

- **UML Profiles and reusable element library**: The CONTSYS Actors library and the two UML profiles demonstrated to improve model quality in the design of the Reusable Web Services
- **ModelHealth Toolchain**: The toolchain utilities (transformations, templates and traceability services) provided by the MDD concepts are utilized for creating UML profiles and the library with reusable model elements. The results demonstrated that the ModelHealth Toolchain was suitable for creating reusable web services in a continuity of care environment.
- **Web Services**: Is the foundation for the design of reusable web services for the homecare domain. The underlying SOA concept facilitates design of Independent Living systems.

This section will discuss the results in terms of the research questions, the findings, and existing knowledge in the field. A discussion on the experiences using the Design Science Research framework is given at the end, commenting on the three research cycles.

5.1 R1 How can a model-driven development toolchain with domain support assist the development of reusable domain software services?

During three iterations in the *"design cycle"* the ModelHealth toolchain was refined based on assessment feedback from both students and professionals. The initial ideas for the development tool support as presented in Paper 1 were implemented in an exploratory manner, starting out with functionality and processes assumed to be practical based on existing knowledge. Version 1 of the toolchain assumed that a MDA compliant UML tool with default configuration would be sufficient for developers to do model-driven development, if they were given appropriate training and proper written documentation. This turned out to be an optimistic assumption, both with respect to students and professionals. The MDD tool had to be adapted to the needs of the developers in order to be perceived useful in developing healthcare software services.

5.1.1 Needs for tool support in model design

The initial student experiment with ModelHealth Toolchain version 1 indicated that the toolchain should provide assistance in creating the correct and required models (F4). The students created models that contained duplicate elements and incorrect model structures, indicating that the lectures and the written material (guidelines) given did not provide the necessary support in the design process. The students seemed to understand the approach and toolchain when the lecturer demonstrated it, but still they made mistakes when working on their own. They expressed that it was "too extensive", and it seemed that the "*cognitive distance*" as discussed by Krueger in (Krueger 1992), was too high between the abstract guidelines given in the written documentation, and the required implementation tasks to be carried out in the modelling tool. As a result from this first design/assess cycle, support for project structure and process assistance was developed (see Section 3.3.3).

For Version 2 of the ModelHealth toolchain, the Enterprise Architect UML modelling tool was extended with a template project structure and UML profiles. The evaluation was conducted in the MPOWER project where professional developers used it over a one-year design period to design and develop reusable software services. The developers shared a version controlled model repository that contained a complete use case model, features model, a common information model and packages with services designs. From the developer survey, it was found that ease of use, and correct and complete code generation were important factors for the usefulness of the tool (F2). One of the developers using the toolchain outlined the need for correct code generation: *"Otherwise you can find yourself spending too much time trying to make things work (and doing the required changes manually)."* This finding confirms other studies such as (Anonsen 2005; Mattsson et al. 2007; Trask et al. 2006)

To fix the code generation problem, Version 3 of the ModelHealth toolchain was created, utilizing an external MOFScript transformation engine provided as an Eclipse plugin. Version 3 was used by university students in a project assignment, before it was finally evaluated in a student experiment and a professional developer survey, reported in Paper 10. The evaluations showed that the participants were able to create complete and correct models, and found the process and approach to software design useful and attractive. This indicates that the issues related to both F2 and F4 were solved in the latest version of the ModelHealth toolchain.

The evaluations also identified the need for model validation and verification, stated as F6. The Object Constrain Language (OCL) (Object Management Group (OMG) 2006; Warmer 2003) is a mechanism that can be utilized for this purpose, but it requires the specification of constraints in a UML profile. Examples of such constraints are presented in Paper 4 where domain-specific constraints are specified as invariants. The use of OCL in modelling is a growing area of research where especially performance and scalability issues are addressed. In (Shaikh et al. 2011), the authors provide an evaluation of the most popular UML/OCL tools available. They also describe an approach to overcome the performance problems, but conclude that more research is required. The ModelHealth toolchain did not apply advanced constraints checking, but identified this as an area for improvement.

The scenarios implemented in the student evaluations were relatively simple, addressing only one system and three stakeholder groups (patient, relatives and healthcare professionals). As the systems become larger, the complexity will increase considerably. In the final student experiment, improved overview and system understanding were outlined as important features. This is an inherit feature of MDD if the design process is carried out in a proper way. System complexity and the use of UML were investigated in 2005, when Arisholm et al published the results from a controlled experiment with students using UML in systems design (Arisholm et al. 2006). They found that in terms of design correctness *"both experiments show that, for the most complex task, UML subjects perform significantly better than no-UML subjects."* It is reason to believe that system design for continuity of care, or healthcare in general could benefit from a MDD design and development approach with proper tool support.

The findings discussed (F2, F4 and F6) address the design and functionality of the MDD tool or toolchain itself, and can be viewed in light of the summary of CASE

tools given by Iivari (Iivari 1996). In his seminal paper, Iivari summarizes the use of CASE tools in industry in 1996. His survey on factors affecting the adoption of CASE tools found that in overall, the CASE tools tend to improve quality of developed systems, and to some degree also productivity. Two of the factors relevant for the work presented herein are that *"relative advantage"* has a positive impact, and that "*perceived tool complexity*" has a negative impact on CASE tool productivity and quality effects. The relative advantage was mainly measured in terms of increased speed, quality, ease and effectiveness of respondents' tasks. Iivari explains the negative impact from "perceived complexity" with "*CASE tools are often complex and when their complexity is perceived to be high it is difficult to appreciate their advantages*." As a result of this finding, Iivari states, "*any means to affect these perceptions [complexity] can be expected to be significant in the management of CASE adoption*."

Staron confirmed the importance of modelling tools maturity in 2006 (Staron 2006). In his case study of MDD adoption in two companies, he concluded that the availability of mature modelling tools is the most important factor for adoption.

5.1.2 Traceability in MDD should be utilized

Most respondents involved in the ModelHealth toolchain investigations highlight the utility of traceability in the toolchain (F3). The toolchain provided a lightweight implementation of the core services associated with traceability; navigation, coverage/orphan analyses and change impact analyses. The respondents found that traceability had a positive impact on system overview and understanding, connecting scenario descriptions, use cases, features, information model and service designs.

Traceability should be utilized for validation of medical software, and in Paper 10, the ModelHealth approach to traceability is discussed in terms of FDA recommendations. The approach builds upon the profiles presented in Paper 4, using visual notations to present traceability analysis results. The proposed solution was not evaluated with developers in the PhD project, but address a core challenge identified by Winkler and Pilgrim:

"*when it comes to using—and particularly visualizing—traceability links, current tools provide no support at all, and consequently, traceability links can often not be put to use."(Winkler,Pilgrim 2009).*

Traceability in MDD has become a central topic in MDD research, evolving from requirements engineering to encompass all artefacts in the development process [ibid.]. However, Winkler and Pilgrim found that "*traceability methods are not used in practice as much as they could. One of the main reasons is lack of good tool support"[ibid.]* The survey also identifies several other challenges such as *recording* trace information, *sharing* trace information between tools and *maintenance* of trace information. With respect to traceability information recording, the experiment participants explicitly stated that benefits of using trace information largely compensated for the effort required to manually create tracelinks. Sharing and maintaining trace information will require substantial changes to the tools in terms of standardization (Limón,Garbajosa 2005), repository creation and interfacing.

The traceability solution presented in Paper 5 address the challenges presented by Winkler and Pilgrim. Introducing a *TraceRepository* for storing information and a *TraceModel* defining an extendable metamodel, the solution supports recording, sharing and maintenance of trace information. If a MDD toolchain incorporates a domain specific and interchangeable *Trace Model*, this would allow for better traceability analyses (Paper 5). The solution presented in Paper 5 was not fully implemented in the ModelHealth toolchain, but is a contribution to future design and development of shared traceability solutions for MDD.

5.1.3 The "Perfect tool¨

There is, and has been a plethora of MDD tools available targeting different domains and user groups. Kleppe defines a set of requirements for MDD tool in (Kleppe 2003) that are used as basis for the *"Perfect Tool"* by MacDonald et al. (MacDonald et al. 2005). The results presented in this PhD project nuance and extend these requirements. To summarize, Figure 30 shows a diagram where the Findings 2, 3, 4 and 6 from the ModelHealth toolchain evaluation extend relevant requirements from Kleppe and MacDonald.

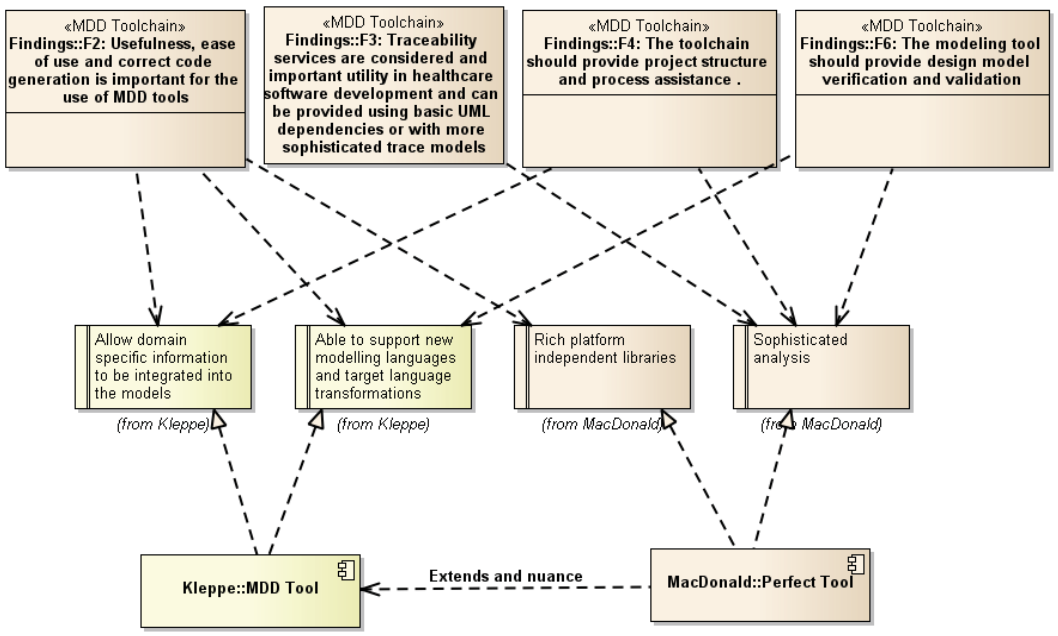

Figure 30 The toolchain findings extending and nuancing the work by Kleppe and MacDonald. Only the relevant requirements from Kleppe and MacDonald are included in the diagram

To which degree the findings can be realized in "one perfect tool" may vary. If one tool realizes all recommended features, it is a danger that it will become too complex and consequently violate one of the most important requirements, namely ease of use.

A software "plugin" framework such as the one provided by Eclipse could be a viable solution for the perfect tool. Using Eclipse, a tool can be composed of a selection of plugins forming a complete development environment supporting MDD. The most

popular open source plugin for Eclipse is Papyrus[11] that became an official project in the Eclipse Helios release in version 3.6 (June 2010). Papyrus is an open source project that aims to be 100% aligned with the UML specifications from OMG. However, the Papyrus tool is not yet complete and version 0.9 was released in with Eclipse 4.2 (Juno) in June 2012.

The design science approach to developing the ModelHealth toolchain required three iterations to reach an acceptable level of quality. Although the ModelHealth toolchain became useful, there is still some way to go for the "perfect tool". In 2009, Mohagheghi et al published a paper on MDE adoption in industry using experience from two large European companies. Their main conclusion is that there it is a challenge providing a suitable domain language to both technical and non-technical personnel, and that there "*is no tool chain at the moment and companies must integrate several tools and perform adaptation themselves*" (Mohagheghi et al. 2009b). This is a resource demanding process that requires both special modelling skills and time. It seems that the "perfect" tool for MDE is still to be developed.

5.2 R2 How can Relevant Domain Standards and Knowledge be Incorporated into a Model-Driven Development Toolchain, and what Aid can they provide in the Design and Development Process?

There are many standards, and versions of these, in the healthcare informatics field. In this PhD project focus has been on standards related to Continuity of Care such as the CONTSYS standard from CEN/ISO (CEN TC251 2006; ISO TC 215 under development). The ISO development version of the standard "*provides a clear conceptual framework to establish the terms of reference of health information systems. The system of concept as well as the process and workflow descriptions are meant as tools for the development of information systems"(ISO TC 215 under development)*. The two findings related to RQ2 build upon the experience from this single standard, but other and similar standards could replace or complement CONTSYS. The findings are discussed in the following subsections.

5.2.1 Continuity of care standard can be incorporated into a UML profile

The ModelHealth Toolchain incorporated the domain knowledge as UML profiles. Knowledge can be expressed as a modelling language extension and a set of reusable modelling elements (Selic 2007). The CONTSYS standard defines a UML model with the core actors and roles in continuity of care such as "HealthcareProfessional", "SubjectOfCare" and "HealthcareOrganization". As presented in Paper 6, these actors are represented as generic reusable model elements. Based on a rigorous process described in the paper, the actor model was extended with specialized actors from the homecare domain, e.g, visting nurse and system components such medication plan.

[11] Papyus webpage on the Eclipse website: http://www.eclipse.org/modeling/mdt/papyrus/

The results from the experiment in Paper 10 found that the students managed to reuse the correct elements from the library based on a scenario description. In (Krueger 1992), Krüger states that *"for a software reuse technique to be effective, it must reduce the cognitive distance between the initial concept of a system and its final executable implementation."* The specialization of actor modelling elements towards the homecare domain reduces the *"cognitive distance"* between the generic concepts defined in CONTSYS and the real actors in the domain. One of the students in the experiment says about the actors library that: "*I think it simplified the process. We found many English terms for what we were looking for"*. The statement confirms that the elements can be reused, hence avoiding conflicts and errors in domain modelling.

To create a complete design of the software services for continuity of care, the Software Services UML Profile from IBM (Johnston 2005) was incorporated and utilized in the ModelHealth template diagrams. Evaluations with students and professionals showed that the profiles were useful and assisted them in creating correct and complete design models. Since the creation of the ModelHealth toolchain, OMG has defined the SOAML standard with a comprehensive UML Profile for modelling service oriented architectures called SoaML (Object Management Group 2009). SoaML was not implemented in the ModelHealth Toolchain as the original modelling process and UML profile was considered sufficient for the scope of use for the ModelHealth toolchain. The SoaML UML Profile is now available for use in many UML modelling tools[12].

Paper 4 describes the development of two UML profiles for the ModelHealth toolchain. The students did not use the full version of two profiles presented in the paper, as they were considered too advanced and not required by the experiment scenario. The profiles were however demonstrated for professional developers of healthcare information systems (Paper 10) who found the toolchain and approach relevant and interesting.

The main difference from the profiles used by the students is that the advanced profiles utilize stereotypes, tagged values and constraints to increase the expressiveness of the modelling language when designing systems for homecare. A "virtual metamodel extension, restrictive" as described by Staron (Staron 2005), is a stereotype that adds a domain specific icon such as a picture of a nurse to the modelling element, together with a well known domain specific label such as "HealthcareProfessional". The Homecare SOA UML Profile includes elements from the "Code generation, restrictive" category. These stereotypes can improve code generation by providing domain information so that code generation scripts can create high-quality code. Extending the modelling language with UML profiles may increase the complexity of the tool, and hence be counterproductive. Tool complexity has been one of the main obstacles to MDD/CASE uptake, and extensions should be considered carefully before being introduced.

[12] SoaML tool support webpage: http://www.omgwiki.org/SoaML/doku.php?id=tool_support

Relevant and potentially useful extensions to the UML profiles are presented in section 5.7

5.2.2 Presentation of domain information in MDD tools

From the evaluations it is recommended that domain information should be an integral part of a modelling tool. However, as the ModelHealth Toolchain did only address one aspect, continuity of care, it is necessary to investigate how other standards or variants of the same standard could be supported. From the evaluations with students it was inferred that the presentation of domain information should be flexible and easy to use (Finding 5). A developer may have preferences for a language extension type, icon or a specific structure of model elements in the tool. The underlying model of the standard must be the same, but the presentation of the model concepts could be adapted to the users' preferences. This finding is supported by Purchase et al in (Purchase et al. 2002) where they conclude that "*choices need to be made regarding which notation to use between semantically equivalent variations within the UML standard. Choosing the variation that most supports the users' comprehension can only enhance the value of the tool or text: empirical studies can assist in determining which of the variations are more suitable.*"

Recently, Ricca did a series of experiments on stereotyped diagrams for web application using subjects with different education level where the results "*indicate that, although, in general, it is not possible to observe any significant benefit associated with the usage of stereotyped diagrams, the availability of stereotypes reduces the gap between subjects with low skill or experience and highly skilled or experienced subjects. Results suggest that organizations employing developers with low experience can achieve a significant performance improvement by adopting stereotyped UML diagrams for Web applications (Ricca et al. 2010).*" The target user group for the ModelHealth toolchain was inexperienced MDD tool users, a user group that could benefit from stereotyped diagrams.

Using UML profiles, the model element presentation variation is mainly limited to element name or label, icon, stereotype and package structure. If these mechanisms are utilized correctly, positive effects similar to those in the ModelHealth toolchain evaluation can be achieved. For instance, Kuzniarz, Staron and Wohlin report from an experiment where the use of stereotypes had a positive effect on model comprehension (Kuzniarz et al. 2004).

5.3 R3 Which Reusable Software Services are Relevant in the Care and Management of Elderly Living in their Homes?

The work on scenario development in the MPOWER project resulted in 25 software services designed with full traceability in the ModelHealth toolchain. The services were developed utilized to develop two proof-of-concept applications. The applications clearly reflect the functionality of the underlying services (see Figure 27 on page 73) demonstrating that the services support application design and development.

This section will discuss the software services in terms of domain needs and software service reuse.

5.3.1 Generic domain services can implement many domain needs

Finding 7 is derived from the comprehensive work on domain needs that led to the specification of 25 software services through a rigorous design process. These services were found to be useful during pilot application development and evaluation, and indicates that definition and implementation of reusable software services is both feasible and beneficial.

The validity of the services is based on both the design process and the pilot system evaluations. A total of 143 persons (family carers, dementia experts, patients and care providers) from four European countries participated in the user needs investigation in the context of the MPOWER project (see table 2 in Paper 9).

The scenarios overlap largely with the BRAID project's scenarios for ICT and active aging (Luis M. Camarinha-Matos et al. 2011), especially with the "Independent Living" area. The BRAID scenarios have been developed and rigorously validated by the leading experts in Europe in the field of aging and independent living. This implies that because of the overlap and similarity with BRAID, the scenarios from which the 25 services are derived are representative for the domain. However, a main difference between the two scenario descriptions is that the BRAID scenarios are not applied in system specifications or service designs, but influence the definition of four ICT related actions in the final report (BRAID Project 2012). The actions can be summarized as:

1. Identify and promote standards in order to facilitate wider take-up, interoperability and affordability of solutions.

2. Develop theoretical foundation for ICT and Ageing and promote a consolidation of concepts and common ontologies.

3. Identify and promote technological development synergies between ICT and Ageing and other focus areas, e.g. construction and building procedures, intelligent transport systems, smart grid infrastructures development, etc.

4. Promote participatory design: Identify suitable approaches and promote pilot experiments on the involvement of seniors in the processes of co-designing systems for ICT and Ageing.

The work on reusable services and application development in this PhD project (Papers 6, 7 and 9) is inline with actions 1, 2 and 4 that clearly focus on standardization, interoperability, consolidation and user-driven development.

Like BRAID, most standards developing organizations (SDOs) don't provide open source reference implementations at the service or application levels of their standards, but share written documents often containing UML diagrams for information and interaction models. The survey with professional healthcare information systems developers reported in Paper 10 found that standards are

important for development, they are hard to read and require interpretation before they are used. A properly documented detailed design or reference implementation of a standards-based component or service would resolve some of the challenges of implementing a standard.

As presented in section 3.5.2 and 4.3.2 the services were developed and organised in five categories covering both general and domain specific functionality. The results from using the services in the implementation of the pilot systems showed that the services are useful for system realisations including information centric features and to some degree domotic sensors. While the results from using the services in the pilot system development are promising, the development of additional systems based on the services would add to the confidence in the reusability of the services.

5.3.2 Access to service design rationale

The design rationale for the 25 reusable software services proposed is part of the UML design model. This rather comprehensive UML model describes the domain actors, use cases with scenario descriptions, and the features derived from these. From these features, each service is completely defined with operations and parameters, including full trace information back to the features and use cases motivating their design. The availability of design rationale is considered an asset in software development. An observation study done in 2007 of developers in Microsoft discovered that the developers spent much resources on finding information about the background for their code: "*The most often deferred searches included knowledge about design and program behavior, such as why code was written a particular way, what a program was supposed to do, and the cause of a program state*" (Ko et al. 2007). The authors suggest further research into what and how this knowledge could be made readily available though the developer tools.

A UML design model having both design and design rationale is a mechanism to reduce the *"cognitive distance"* between a requirements specification and the actual system implementation (Krueger 1992). The approach can assist developers in understanding and remembering how the services were related to the user needs and be useful for understanding and comparing the domain user needs and services to those from other projects and standards. An important aspect of this approach is that domain experts can provide important contribution to the rationale models – scenario and use cases, which has a positive impact on the model validity and quality. In (Lenz et al. 2007), Lenz et al argue for a separation of domain concepts and system implementation:

"in order to cope with domain evolution, modelling of domain concepts should be separated from IT system implementation. IT systems should be implemented by IT experts and medical knowledge should be modelled and maintained by domain experts."

In addition to requirements specification and domain best design practices, healthcare software developers must implement software that adheres to a set of domain standards. The results from the professional developer survey (Paper 10) shows that standards relevant for systems development are of utmost importance, but can be hard to understand and require interpretation before they can be applied. In some cases,

interpretations done by different developers may lead to interoperability problems between systems and / or components using the same standard.

To overcome the standards interpretation problem, events like the Continua Alliance Plugfest [13] and Integrating Healthcare Enterprise Connectathon [14] are organized annually. At these events healthcare IT industry can meet for performing interoperability testing and go through a certification program. To facilitate software reuse, the Continua Alliance maintains a repository of validated implementations of their standards along with a test tool. These resources are only available to their paying members[15]. The initiatives by Continua and IHE are positive, and they confirm the need for improving the way standards can be utilized by software developers today. Reusing open source software is not easy and there are many pitfalls that must be avoided (Spinellis,Szyperski 2004). The process of reusing software components and code is outside the main scope of this thesis and is not discussed in detail.

5.3.3 Application development from basic services

Finding 8 is based on the design and development experience presented in paper 7 and 9 where it was shown that the Norwegian MPOWER pilot system improved the everyday life of the elderly and their family. In addition, the care providers (visiting nurses) reported that they saved time during visits because the elderly users were aware of and prepared for their visits. The Norwegian system was developed by applying a participatory design process and was introduced to the users gradually. In addition to user-friendliness (Demiris et al. 2004), proper training of careers, frequent visits by the nurses in the initial phase, user training sessions and family motivation were critical factors for system adoption and use (Paper 7).

The Norwegian pilot system was designed for information sharing, primarily using a shared calendar and a messaging service. Both these functions were developed from the reusable software services, including the required security services (authentication and authorization) and patient management services (relationships, contact information etc.)

One advantage of the service-oriented architecture in this case is that the system through rapid development comparatively easy can be tailored into similar systems, concerning the GUI or the interface in general to the user. In case there is a need for services beyond those provided as reusable components, adding new services to the application platform is easy.

The experience from pilot systems development substantiate that a rapid development of applications for different kind of user groups or single end-users is possible (Paper

[13] Continua Alliance Plugfest homepage: http://plugfest.continuaalliance.org/

[14] IHE Connectathon homepage: http://www.ihe.net/Connectathon/index.cfm

[15] Continua Reference Code libary: http://members.continuaalliance.org/continua_align/products_services/

9). For systems addressing continuity of care and independent living this could be even more important than for other domains. The end users in the domain would benefit from a flexible tailoring of the systems to their individual capabilities and needs (e.g. user interface personalization and configuration of sensors and actuators for the home).

The answer to RQ3 is given by Finding 7 and 8, stating that a limited number of reusable services can be utilized in the development of useful SOA-based applications supporting continuity of care. The overlap between the rationale for the 25 services and the BRAID scenarios indicates that it is reason to believe that the services can be utilized for other types of applications in the domain.

5.4 Design Science Experience

As presented in section 3.1, the research followed a design science approach where two main artefacts where developed and evaluated: the ModelHealth toolchain and the set of reusable software services. The Design Science research framework (Hevner et al. 2004) prescribes that the results produced in the design cycle (Hevner 2007) should be made available as useful artefacts in the target domain (relevance cycle) and new knowledge (rigor cycle).

The Design Science framework presented in Figure 1 on page 8 was instantiated as shown in Figure 31. The figure shows activities, results and relationships between these for all three phases (see 3.2). The activities and results are placed in the associated framework cycle to illustrate how the artefact development and assessment activities are influenced by both rigor and relevance.

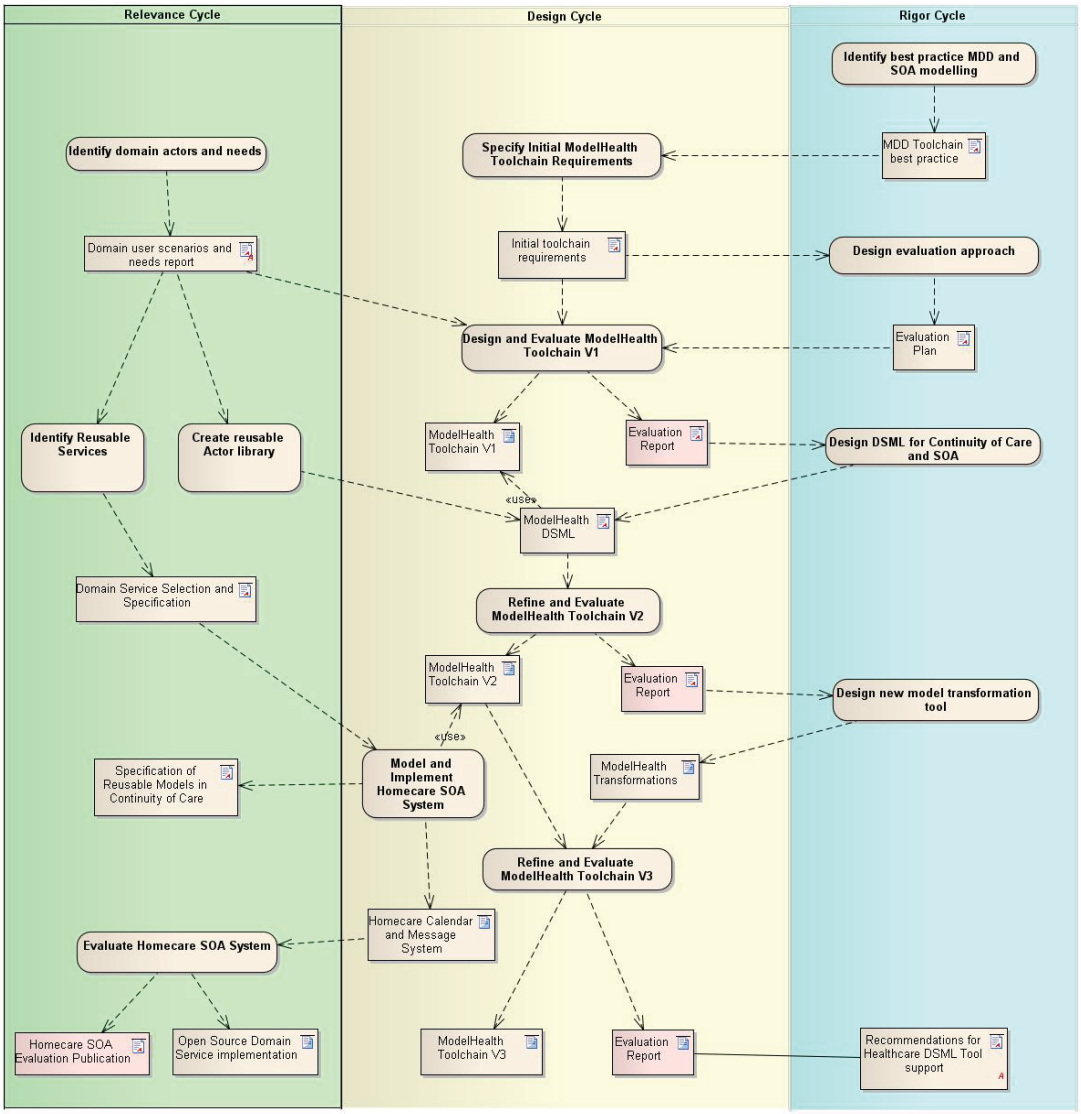

Figure 31 The design science process followed in the PhD project

In the proceeding subsections the PhD project work is discussed in terms of these three research cycles of design science.

5.4.1 Relevance cycle

The relevance cycle must address *"**opportunities** and problems in an actual **application environment**."* (Hevner 2007; Iivari 2007). In this thesis, the problem of providing continuity of care in an independent living situation was addressed with two artefacts representing opportunities in the domain: a model-driven development toolchain and a set of reusable software services.

Both artefacts target developers of SOA-based software in the healthcare domain; an ***"application environment"*** experiencing increased attention as a result of digitalization and need for information integration in healthcare. The integration process is both complex and expensive, and despite numerous standardization

initiatives and reference projects, system interoperability on the information level remains a major challenge in health information systems design and implementation.

Information and communication standards are important to develop interoperable systems, and sharing the same understanding of the underlying concepts is fundamental. An ***"opportunity"*** to address this goal was identified in the use of MDD with DSML support to create domain specific reusable SOA services. The initial requirements and first version of the ModelHealth toolchain was developed with software developers from Croatia, Cyprus, Spain, Austria and Norway, a process that also involved 143 stakeholders from the healthcare domain. The toolchain allows for alignment and reuse of architectural design (patterns) such as actor-component relationships, service descriptions and information models. New requirements were identified in the toolchain design cycle process as described in sections 3.2 and 3.3.

Version 2 of the toolchain was used to design the set of reusable software services based on the results from a previous activity in the relevance cycle. The services that were created were reused in two pilot systems as described in Paper 9. The experience from the service design process resulted in a refined version of the toolchain that was subject to a thorough investigation with both students and professional developers.

To complete the relevance cycle it is necessary to address if *"…the design artefact improve the environment and how can this improvement be measured? The output from the design science research must be returned into the environment for study and evaluation in the application domain."(Hevner 2007).*

- The ModelHealth toolchain was not fully studied in the target application domain due to time and resource limitations. A "technology transfer" to a software company will require substantial investment from the company's side, as well as a well-defined methodology for evaluating its utility. Yet, the findings from the ModelHealth Toolchain design cycle may have relevance implications.
- The ModelHealth toolchain was evaluated by professional developers in the healthcare domain as presented in Paper 10.
- The pilot systems were evaluated with real life settings. The Norwegian pilot system trial period was one year and was considered successful. Ideally, more systems and new variants of the systems should be created and evaluated to get more knowledge about the relevance of the reusable services.

Relevant for both artefacts is that the assessments revealed new requirements for tools and services in the care domain, a result that Hevner also identifies: "*Another result of field testing may be that the requirements input to the design science research were incorrect or incomplete with the resulting artifact satisfying the requirement but still inadequate to the problem presented"(Hevner 2007).*

5.4.2 Rigor Cycle - Research contribution

The rigor cycle should ensure that the artefact represent real innovation and should "*guarantee that the designs produced are research contribution and not routine designs based upon the application of well-known processes*" (Hevner et al. 2004). Important factors that were input to the artefacts design processes were:

- Experience from and knowledge about other MDD frameworks that have not reached their potential in terms of use in domains such as healthcare. Many of these have failed due to poor user friendliness and high complexity. Much effort was made in keeping the ModelHealth toolchain as easy as possible, "restricting" the size and freedom of UML, and providing domain support in the required design steps. The results indicate that the ModelHealth toolchain was quite successful in doing this.
- Knowledge about the trends towards information integration in the healthcare domain using web-based portal solutions. Service-orientation has already proved to be feasible in other domains such as travel and finance. Healthcare enterprises made strategies where SOA was a core approach to interoperable systems.

To complete the rigor cycle the ultimate assessment is: "*What is new and interesting contributions?*" This can be answered with three types of contributions to the knowledge base:

- The ModelHealth toolchain artefact: The ModelHealth toolchain provides one solution to developing healthcare specific web services that are based on a proper design and conform to selected healthcare standards. This extends the "knowledge base" with experience about the use of UML Profiles and reusable model element libraries in an UML-based MDD toolchain. Non-expert developers can, with the aid of a domain specific MDD toolchain, design and develop SOA software with decent quality. In his *"Epistemology of Design Science",* Iivari defines this as a prescriptive research contribution (Iivari 2007) where the artefact demonstrates a possible instantiation, but has no truth value.
- Foundations: Traceability in MDD has gained much attention recently, broadening the focus from requirements and test to a complete traceability scheme that includes all development artefacts. Several tool vendors have implemented explicit traceability service functionality, but not much is known about the real use of these. The results from the experiment presented herein show that a basic implementation of the three traceability services trace navigation, coverage and orphan analysis is found useful and easy to use. The professional developer survey confirms the necessity for traceability services.
- Reusable software service designs and open source reference implementations: The service designs approach where each service interface operation can be traced back to an user-specified design has received much attention in conference and workshop presentations, and is clearly a contribution to the knowledge base. Furthermore is the experience from using the services in systems development a valuable case study which other developers and researchers can learn from.

5.4.3 Design Cycle

The design cycle includes three cycles of toolchain design, and one long cycle with software service design:

- The initial toolchain was designed to fulfill the requirements specified by professional developer in the MPOWER project (see Paper 10.) Following the "design as a search process" approach, the ModelHealth Toolchain was assessed and refined three times before the final evaluations. The initial assessments were conducted on different subjects using different evaluation techniques.
- The definition of the services was done using the user needs specification and best practice SOA design principles as input. The process diagram in Figure 31 shows only one iteration where the services were evaluated in the pilot system development and evaluation. Minor updates to the services where done during pilot system development, but was not considered a complete design cycle.

The main challenge within the design cycle was as declared by Hevner, to *balance the efforts* spent in constructing and evaluating the artefact. In the case of toolchain design, it was clear already after the first student evaluation that the toolchain needed to provide modelling support on both structure and content. The "Actors library" was designed and specified in a systematic process to ensure domain compatibility (Walderhaug et al. 2008c). The following evaluation demonstrated some utility, but the transformation mechanism required a redesign (Walderhaug et al. 2008a). The transformation component redesign required significant effort (3-400 hours including testing) in developing the transformation stylesheet that allowed for export from Sparx Enterprise Architect to Eclipse for use with the MOFScript plugin. The assess-refine cycle from version 2 to 3 was longer than initially planned, but proved to be worthwhile as the transformation component worked as intended in the following evaluations.

In the case of software services design, the complete define / assess process required much time as it was deemed necessary to test in realistic environments, with real users and real environments. The application development and evaluation was mainly carried out by project partners in the MPOWER project (see section 3.6). The recruitment process of real users, training of carers (informal and formal) and setting up the required hardware, took more time than initially planned. As a result, only one major design cycle was completed. Minor service refinements such as adding parameters or changing operation names were done as a part of the implementation process and was not considered design cycles.

As for most development projects, the resources available restricted the choice of evaluation methods. To keep the design cycles (assess-refine) as short as possible, the "effort-balancing problem" allowed for relatively short and focused studies of the toolchain. The evaluation methods and data collection used a combination of direct and indirect techniques defined in the taxonomy from Singer et al. (Sim,Lethbridge 2008). As discussed in Paper 10 the low number of participants is a major threat to validity, for both the student experiment and the processional developer survey. On the other hand, a total of 28 students and 41 professional developers took part in the overall *design cycle* from 2008-2010, implying that the results should be a valid basis for further investigations, preferably in the "target application domain".

It can also be seen as a necessity to do smaller investigations in areas where it doesn't exist much evidence. In (Arisholm et al. 2006), Arisholm et al concludes on their

experimental evaluation of UML documentation: "*In terms of experimental methodology, we have found it very useful to start with a smaller experiment and dwell on qualitative analysis at first. This has allowed us to better understand what issues might come up in subsequent, larger experiments. Based on the first experiment we decided, for example, to use more complex change tasks in the second experiment*."

In terms of Design Science, never reaching the "technology transfer" phase is not uncommon, and as Iivari says in (Iivari 2007) "*...artefacts developed in design science should first be tested in laboratory and experimental situations as far as possible. One should not start with testing in the real situations, except perhaps in very exceptional special situations*".

5.5 Implications for research and practices

The results from the work conducted in this project may have implications for research and practices in the field of healthcare specific model driven development, and in some cases model-driven development in general. In this section, the potential implications are discussed in terms of the findings and the research approach. Some of the implications are also discussed in Paper 9 and 10.

5.5.1 Domain specific model-driven development

During the last decade, there seems to be an increased focus on domain specific MDD. In 2007, France and Rumpe outlined a research roadmap for model-driven development (engineering) of complex systems where they conclude that: "*In the MDE vision, domain architects will be able to produce domain specific application development environments (DSAEs). Software developers will use DSAEs to produce and evolve members of an application family*". A DSAE consists of tools to *"create, evolve, analyze, and transform models to forms from which implementation, deployment and runtime artifacts can be generated. Models are stored in a repository that tracks relationships across modeled concepts and maintains metadata on the manipulations that are performed on models*."(France,Rumpe 2007)

Findings 1-6 presented in this thesis indicate that the "perfect" tool or DSAE can or should be many tools or a toolchain, tailored to the users' preferences. The degree of tool perfection depends on the user and the target application domain. For the healthcare domain, the requirements related to information coding and representation will have a high priority due to the interoperability challenges discussed. In this sense, the approach to tool development followed by the OpenHealthTools project[16] seems promising. They see the tool as an essential part of an *ecosystem* and their technical goal *"is to assemble and/or develop a comprehensive harmonized tool suite to enable the definition, development and deployment of interoperable Electronic Health Records."* To manage this, they have defined a set of architecture principles that focus strongly on "user needs", "good enough" and "working code".

[16] OpenHealthTools website: http://openhealthtools.org/

Creating ecosystems could be a DSAE and represents a new approach to artefact development where several tools and systems support the complete artefact lifecycle, including design, development, evaluation, deployment and maintenance. Development of a healthcare DSAE should consider the research presented in this thesis about MDD tools and domain specialization with recommendations on which utilities that are considered useful and how these should be provided.

The results in this thesis addressing MDD toolchain design and domain standard incorporation into tools can provide valuable input to the OpenHealthTool project and similar initiatives. Findings 1-6 all give relevant input to how tools and ecosystem components should be designed. Moreover, the design science experience should motivate tool researchers and developers to maintain a strong focus on all three cycles: relevance, design and rigor.

Based on Findings 1-6, the recommendation for practices ***with respect to domain specific MDD is to explicitly consider how to present and integrate utilities such as domain libraries, traceability and transformation, as part of a complete MDD toolchain with focus on usefulness, efficacy and ease of use.***

5.5.2 Reusable SOA services supporting independent living and continuity of care

Haux argues that it is necessary to explore new architectural styles to support trans-institutional information sharing in healthcare (Haux 2004). The development of systems to support independent living and continuity of care involves many institutions and domains. Services supporting independent living do not and will not exist in isolation and the domain consists of a large set of independently developed systems and services. The European commission, the NHIN CONNECT in USA, standards development organizations such as HL7, as well as other national initiatives have launched large programs for standardizing information sharing in the healthcare sector, many of which focus on SOA-based platforms. These platforms include "core services" for e.g., addressing and security. It is reason to believe that the core services will be extended with services closer to the application domain.

Findings 7 and 8 states that a relatively small number of reusable software services can be utilized to create powerful homecare applications. The software services (see section 4.3.1) are based on SOA by using the web services WSDL and SOAP specifications. These are interoperability enablers, as the web service front end allows heterogeneous platforms to interoperate (e.g. .NET and Java.). However, the main challenge lies in developing reusable service designs. The methodology and tools for designing the services described in section 3.4, as well as the SOA platform employed in the pilot systems, may be useful to other designers and developers in care domain. The services and the tools are provided as open source, and the referenced publications document the methodology and lessons learned.

Some standards development organizations such as HL7 are working on standardizing core software services. The Healthcare Service Specification Project (HSSP) is a joint initiative by OMG and HL7 working towards definition and reference implementations of reusable software services for healthcare. Since 2006, the HSSP

group has slowly grown in number of subprojects and partners. In June 2012, a project called HSSP Care Coordination Service was initiated[17]. This is a first step towards standardization of services that are relevant for independent living, and it is reason to believe that other SDOs will follow with specifications. The experience from service design and application development presented in this thesis is relevant for this standardization work.

Based on Findings 7 and 8, the recommendation for practices in this area is to ***work towards standardization of functional and reusable domain service designs and verify these through reference implementations and application development.***

5.5.3 Design Science in healthcare informatics

The experience from applying a design science as described by Hevner (Hevner et al. 2004) was positive, and the explicit focus on relevance, assessment cycles and rigor fits research in the healthcare informatics field perfectly. The characteristics of the research domain with a plethora of stakeholders, systems, concerns and standards, imply that a successful artefact should undergo repeated evaluations in realistic environments using sound methodologies and foundations. The growing need and provision of "ambient assistive living" technologies will require service personalization and need requirements for quality in use (Walderhaug et al. 2012). The experience from the work on this thesis is that Design science provides a powerful research framework for innovative artefact development in this domain.

As previously pointed out, only one design cycle was carried out for the development of reusable software services. The process of evaluating software systems with real users in real environments requires significant resources. For some development projects this is not possible, and simplifications in the evaluation process must be made. This will be an unfortunate situation that can be improved by the development of "design guidelines" similar to those provided for healthcare specific MDD in Paper 10. Design guidelines are provided by several standard development organizations, e.g. IHE Profiles (IHE 2012) and HSSP's SOA4HL7 methodology (Honey,Lund 2006).

From the experience in this PhD project, the recommendation for practices is to ***apply design science for artefact development in the healthcare informatics domain, and stress to keep the assess/refine cycles as short as possible.***

5.6 Limitations and threats to validity

The scope of the research presented in this thesis was limited by several factors that were outside my control. Consequently, in order to get useful and valid research results, the scope was carefully specified by:

- Target application domain

[17] HSSP Care Coordination Service: http://hssp-carecoordination.wikispaces.com/home

    - The software services should be applicable in the area of home care / independent living, addressing both social and healthcare needs. The target users are in general healthy people suffering from mild dementia or similar cognitive impairments. The ModelHealth toolchain addresses novice and medium skilled software developers working in the healthcare information system area. They have no or medium experience in model-driven development, but should have a basic understanding of UML modelling. To increase adoption of MDD, it was considered important to address the needs of non-expert MDD users.
- Resources available in the project period
    - Getting access to caregivers and elderly is a time-consuming activity that often restricts research having a strong focus on engineering. As this research was partly conducted as a part of the MPOWER project, access to a large group of stakeholders from four European countries was made possible. The selection of these stakeholders was part of the MPOWER project plan.
    - Getting access to software developers is hard and could require special funding. The MPOWER project allowed for using professional developers within the project as evaluation subjects. Furthermore, students at the department for computer science at the University of Tromsø, Norway were available for experiments through specific courses taught at the institute. The number of students participating was limited by the course.
    - Time was considered a limitation in both toolchain design and evaluation. This classical trade-off between technical perfection and evaluation rigor is discussed later.

These limitations influenced the choice of research approach, and the selection of evaluation methods were decided according to the guidelines given by (Easterbrook et al. 2008).

In Paper 7 the weaknesses with the end user trial in Trondheim is discussed. Being a development project, the trial suffers from several shortcomings with respect to explanation of effects. The evaluation conducted face-to-face interviews and explicit statements from the users involved were captured. These are presented in Paper 7 and the positive claims are based on these. The rather long evaluation period by the majority of the users (10 months) improves the validity of the statements.

In Paper 10, a thorough discussion of the internal and external threats to validity with respect to the ModelHealth toolchain design and evaluation is provided. The core factors from this discussion are:

- **Researcher bias:** The author is both the creator and evaluator of the tool, but has no commercial or financial interest in its outcomes. The bias was addressed by using several information sources in the experiment. Moreover, the results were discussed with fellow researchers, and all raw materials are available to the readers on request. Researcher bias is a common problem for such studies. This affects the task description, observation, interviews, coding and interpretation of the results. To address this, the results (models, codes, interview transcripts, and

analysis) are made available for others on request to the author. The results presented are documented with at least two sources of information, e.g., an interview statement that is supported by an observation

- **Low number of subjects in the evaluations:** Ideally, a larger group of developers should be involved in both studies to be able to do more statistical calculations on the material. In the case of the experiment, a larger group could have changed the context from an open dialog between developer and researcher to a more closed relationship where the students would feel stressed and observed. Neither of these factors were mentioned as a problem during the interviews, nor observed as a limitation during the exercise. The professional developer survey was conducted in the largest health information system developer workshop organized in Norway. The EHR developer company organizing the workshop has a large market share and clear strategy on standardization and integration and is the provider of the main integration software used in the between hospitals and primary Norwegian Health Network. The workshop participants were all using or working with the products from the organizers; hence it is reason to believe that they represent the target developer community
- **Lack of control group for toolchain experiment:** The lack of a study control group makes the interpretation of the study results more difficult. A control group and a cross-study design would allow for comparison of methods. However the problem of creating a baseline would create important validity issues with such an approach as discussed by Kitchenham et al in (Kitchenham et al. 2002).
- **Using Students:** Using students as study subjects have been debated profoundly in the field of empirical software engineering (Carver et al. 2003; Jedlitschka,Briand 2007). There are pros and cons, but as stated in the report from the working group on "The role of controlled experiments": "*there is no common agreement on this point*" (Jedlitschka,Briand 2007). Carver et al. conclude that student experiments are useful under proper conditions. They provide guidelines that were followed in this work (Carver et al. 2003).

5.7 Recommendations for future work

As illustrated in Figure 29 on page 75, the scope of the work done in this PhD project is focused on the intersection between independent living and continuity of care specific model-driven development for SOA. Both the development of the ModelHealth Toolchain and the design and appliance of the reusable software services produced findings that should be pursued in future work. In section 5.5, a set of expected implications were discussed. The recommendations for future work given in this section are aligned with these implications:

- Domain specific model-driven development: it is a need for more investigations on how domain libraries can be standardized and easily incorporated into MDD tools.
    - The UML profiles developed in this project should be aligned with ongoing initiatives in healthcare SDOs as well as OMG. The OpenHealthTools and the HSSP projects are existing arenas that could be used as collaboration partners towards standardization of DSML for independent living and continuity of care.

  - More research should be conducted on how to present healthcare knowledge in a coherent and unambiguously way using the limited set of mechanisms available in UML Profiles.
  - Similarly, the software service profile currently used by the ModelHealth Toolchain should clearly be updated with a simplified version of the SoaML UML profile from OMG. It is important that the toolchain ease of use is not compromised.
- Reusable SOA services supporting independent living: even though the services developed in this project were rigorously designed and found useful in two pilot systems, more work should be done on refining their interfaces.
  - Projects such as BRAID have documented domain knowledge that can be utilized to check the "functionality coverage" of the services. It may become necessary to refine the interface messages and add operations to fully support important scenarios.
  - The PhD project created complete service designs in UML for all the identified services. The design methodology as well as the design language should be introduced to SDOs and aligned with their ongoing processes in order to provide reusable service designs that can be imported into the most popular MDD tools for inclusion in service and application development.
- Design science in healthcare informatics: an important part of the PhD project was the focus on the three research cycles: relevance, design and rigor. Despite well-known challenges such as the "effort balancing problem" between design and assessment, the design science framework is considered highly relevant for artefact development in healthcare informatics. Many researchers in the domain follow one or more of the research cycles of the framework, but it seems that few have a clear understanding of this. Based on the experience in this work, design science, with a strong focus on all three cycles, should be presented and discussed as the suggested approach to healthcare informatics research. In these discussions, focus on short cycles and sound evaluation methods should receive much attention.

Chapter 6 Conclusions

The previous chapter discussed the answers to the three research questions posed in Chapter 1. This final chapter presents the conclusions regarding the overall research problem and questions, and lists the contributions made by this thesis.

The overall problem addressed in this thesis was motivated by the challenges associated with IT systems for continuity of care, the increased focus on SOA strategies in healthcare, and the advances in domain specific model-driven development techniques. This led to the following research problem statement:

How can software developers utilize model-driven development to develop reusable software services to support care and management of elderly in a homecare environment?

To focus the investigation, the research problem was addressed by investigation three research questions. Before answering the overall problem, each of the three questions is answered based on the discussions in the previous.

R1: How can a model-driven development toolchain with domain support assist the development of reusable domain software services?

The main conclusion regarding the first research question is that a *MDD toolchain can assist the development of reusable domain software services by providing domain-specific mechanisms such as UML profiles stereotypes and design templates that align the design with domain best practices. The toolchain should also provide mechanisms for model validation and verification to improve the design model quality.*

It was shown with the evolution of the ModelHealth toolchain that default tool configuration did not provide sufficient ease of use and usefulness for the users. Through three design cycles, the toolchain was refined according to the findings in the assessments. The users perceived the final version of the toolchain useful, and the quality of the results produced in the experiment was satisfactory. The toolchain supported the development with:

- Improved understanding of the target system environment by incorporating a homecare UML profile. The UML profile contains a domain specific stereotypes and a library of actor elements based on a standard for continuity of care.

- Facilitating the creation of traceability information during the design process. This allowed for traceability analysis and improved overview of the system under design.
- Simplify the design process by providing project and design templates that improved the design model correctness quality and toolchain ease of use.

In addition, it was found that model verification and validation could improve the design model quality and hence the toolchains usefulness. All the reusable software services were designed and developed with the ModelHealth toolchain, with positive feedback from the developers.

Recent studies have found that MDD tools are still immature and there is no "Perfect tool" available to the developers today. Hence, the MDD tools must be adaptable to different needs in order to provide the required utility.

R2: How can relevant domain standards and knowledge be incorporated into a model-driven development toolchain, and what aid can they provide in the design and development process?

The main conclusion regarding the second research question is that *for standards such as the ISO Continuity of Care, UML profiles can incorporate the core concepts and allow developers to utilize them during design model development. The presentation of this domain knowledge must be carefully designed so that it allows for easy and correct use by the developers.*

A core objective of the ISO Continuity of Care standard is to assist developers in developing interoperable software. The work in this PhD Project demonstrated that the core parts of this standard could be incorporated into a UML Profile with stereotypes and reusable UML actor elements. Furthermore, a model-driven development toolchain can provide design process support that assist developers in creating high quality design models.

The presentation of domain knowledge and concepts from the standards is often limited by the tool's meta language support. In terms of UML, the UML Profile only allows for simple presentation using an icon and a label and it may be a challenge to provide a powerful presentation of complex domain knowledge. The work shows that repetitive user evaluations contribute to improving the users' comprehension of the domain knowledge in the tool.

R3: Which reusable software services are relevant in the care and management of elderly living in their homes?

The main conclusion regarding the third research question is that *services for medical and social information, communication, sensor frameworks, interoperability and security can cover a significant part of the system functionality required to support the most typical user scenarios in the domain. Simple information systems reusing*

these services have the potential to provide important continuity of care functionality such as sharing information between care providers.

Following a rigorous and resource demanding process, a set of 25 services were identified, designed, developed and utilized in pilot system development within the MPOWER project. The identification and design process involved many different stakeholders and was carefully documented through scenarios, UML use cases, feature diagrams and UML software service designs. The services are provided as open source and are grouped in five categories: medical and social information, communication, sensor framework, interoperability and security.

To evaluate the usefulness of the designed services, two pilot systems were created using the services as building blocks. The development process and reuse of the software services were found useful. The evaluation of the Norwegian information-centric information system in a one-year period found that the system was valid and useful.

The validity of the services was only evaluated through development of two systems. Because the service designs can be traced back to the originating scenarios, the domain validity can be compared to other domain knowledge documents such as those provided by the BRAID project.

Overall conclusion

The main contribution of this PhD project is comprised by the answers to the three research questions. The answer to the overall research problem is that developers should be able to utilize model-driven development for developing reusable domain software services. To succeed in doing so, the MDD toolchain should:

- Be easy to use and support development of high quality models and correct model to code transformation.
- Incorporate domain knowledge in terms of stereotypes, reusable model elements, project structures and design templates. The presentation of the knowledge should be possible to adapt to the users' needs.
- Provide a traceability mechanism that allows for full traceability from user scenarios to reusable software service designs and code.

Refinements in the design should be done after sound evaluations in real environments with end users.

Chapter 7 References

Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model Traceability. IBM Systems Journal **45**(3), 515-526 (2006)

Anonsen, S.: Experiences in modeling for a domain specific language. UML Modeling Languages and Applications, 187-197 (2005)

Arisholm, E., Briand, L.C., Hove, S.E., Labiche, Y.: The impact of UML documentation on software maintenance: An experimental evaluation. IEEE Transactions on Software Engineering **32**(6), 365-381 (2006)

Beale, T.: Archetypes: Constraint-based Domain Models for Future-Proof Information Systems. In: OOPSLA 2002 Workshop on behavioural semantics, Portland, Oregon, USA 2002

Bleich, H.H.L., Slack, W.W.V.: Designing a hospital information system: a comparison of interfaced and integrated systems. M.D. computing **9**(5), 293-296 (1992)

Blobel, B.B., Pharow, P.P.: A model-driven approach for the german health telematics architectural framework and the related security infrastructure. Studies in health technology and informatics **116**, 391-396 (2005)

Boehm, B.: A view of 20th and 21st century software engineering. Paper presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China,

BRAID Project: Bridging Research in Ageing and ICT Development. Executive Summary. In: Hadjri, K. (ed.). (2012)

Brailer, D.: Interoperability: the key to the future health care system. Health Affairs Web Exclusive (2005)

Brooks, F.P.: No silver bullet. IEEE Computer **20**(4), 10-19 (1987)

Calliope Network: EHealth Interoperability: State of play and future perspectives. An assessment of European countries' responses to questionnaire on recommendation (COM(2008)594). In. (2008)

CEN TC251: EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Consepts. In., p. 105. European Committee for Standardization, (2006)

Dan, A., Johnson, R.D., Carrato, T.: SOA service reuse by design. In: 2008, pp. 25-28. ACM

Davis, F.D., Bagozzi, R.P., Warshaw, P.R.: User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. Management Science **35**(8), 982-1003 (1989)

Demiris, G., Rantz, M., Aud, M., Marek, K., Tyrer, H., Skubic, M., Hussam, A.: Older adults attitudes towards and perceptions of smart home technologies: a pilot study. Medical Informatics and the Internet in Medicine **29**, 87-94 (2004)

Easterbrook, S., Singer, J., Storey, M., Damian, D.: Selecting empirical methods for software engineering research. Guide to Advanced Empirical Software Engineering, 285-311 (2008)

Eichelberg, M., Aden, T., Riesmeier, J., Dogac, A., Laleci, G.B.: A survey and analysis of Electronic Healthcare Record standards, vol. 37. ACM Press, (2005)

Erl, T.: Service-Oriented Architecture Concepts, Technology, and Design. The Prentice Hall Service-Oriented Computing Series Prentice Hall, Crawfordswille, Indiana, USA (2006)

European Commision: Commission Recommendation of 2 July 2008 on cross-border interoperability of electronic health record systems (notified under document number C(2008) 3282). In., vol. 32008H0594. (2008)

European Commision: The 2009 Ageing Report : Economic and budgetary projections for the EU-27 Member States (2008-2060). In: European Commision (DG ECFIN) and the Economic Policy Committee (AWG) (ed.) 2009 Aging report. European Commision, Brussels (2009)

European Commision: ICT Challenge 5: ICT for Health, Ageing Well, Inclusion and Governance. http://cordis.europa.eu/fp7/ict/programme/challenge5_en.html (2011). Accessed June 10 2011

Finnigan, D., Kemp, E.A., Mehandjiska, D.: Towards an ideal CASE tool. In: Kemp, E.A. (ed.) Software Methods and Tools, 2000. SMT 2000. Proceedings. International Conference on 2000, pp. 189-197

Fowler, M.: UML Modes. http://www.martinfowler.com/bliki/UmlMode.html (2003). Accessed November 9 2012

Fowler, M.: Domain specific languages, 1 ed. Addison-Wesley Professional; , (2010)

France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In, Minneapolis, MN, United States 2007. FoSE 2007: Future of Software Engineering, pp. 37-54. Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States

Fuhrmann, H., von Hanxleden, R., Rennhack, J., Koch, J.: Model-Based System Design of Time-Triggered Architectures - Avionics Case Study. In: 25th Digital Avionics Systems Conference, 2006 IEEE/AIAA, 15-19 Oct. 2006 2006, pp. 1-12

Giuse, D.A., Kuhn, K.A.: Health information systems challenges: the Heidelberg conference and the future. International Journal of Medical Informatics **69**(2-3), 105-114 (2003)

Grimson, J., Grimson, W., Hasselbring, W.: The SI Challenge in Health Care. Commun. ACM **43**(6), 48--55 (2000)

Guttman, M., Parodi, J.: REAL-LIFE MDA: Solving business problems with model driven architecture. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, (2006)

Hagen, I., Cahill, S., Begley, E., Macijauskiene, J., Gilliard, J., Jones, K., Topo, P., Saarikalle, K., Holthe, T., Duff, p.: Assessment of usefulness of assistive technologies for people with dementia. Assistive Technology: From Virtuality to Reality, 348 - 352 (2005)

Haggerty, J., Reid, R., Freeman, G., Starfield, B., Adair, C., McKendry, R.: Continuity of care: a multidisciplinary review. British Medical Journal **327**(7425), 1219 (2003)

Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal **45**(3), 451-461 (2006)

Hartman, A.: Industrial ROI, Assessment, and Feedback - Master Document. In., p. 24. IBM Haifa Research Lab, (2006)

Haux, R.: Health information systems - past, present, future. International Journal of Medical Informatics **75**(3-4), 268-281 (2004)

Hevner, A.R.: A three cycle view of design science research. Scandinavian Journal of Information Systems **19**(2) (2007)

Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in Information Systems research. Mis Quarterly **28**(1), 75-105 (2004)

Honey, A., Dutta, A., Kumar, M., Christian, M.: SOA4HL7 Architecture Document. In: Dutta, A. (ed.). p. 76. Health Level Seven, (2006)

Honey, A., Lund, B.: Service Oriented Architecture and HL7 v3: Methodology. In., p. 79. HL7 Service Oriented Architecture Special Interest Group (SOA SIG), (2006)

HSSP Project: The HSSP Roadmap: HSSP, Version 1.0. In., p. 13. Joint HL7-OMG Healthcare Services Specification Project, (2007)

Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: 2011, pp. 471-480. ACM

IEEE: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. In. IEEE, (2000)

IHE: Integrating the Healthcare Enterprise. http://www.ihe.net (2012). Accessed November 9 2012

Iivari, J.: Why are CASE tools not used? Communications of the ACM **39**(10), 94-103 (1996)

Iivari, J.: A paradigmatic analysis of information systems as a design science. Scandinavian Journal of Information Systems **19**(2), 5 (2007)

ISO TC 215: ISO/CD 13940 Health informatics -- System of concepts to support continuity of care. In: CONTSYS. (under development)

ISO/TC215: Health Informatics - Service Architecture. In: 12967-1/2/3:2009. ANSI, (2009)

Johnston, S.: UML 2.0 Profile for Software Services. http://www.ibm.com/developerworks/rational/library/05/419_soa/ (2005). Accessed November 9 2012

Jones, V., Rensink, A., Brinksma, E.: Modelling mobile health systems: an application of augmented MDA for the extended healthcare enterprise. In: 2005, pp. 58-69

Karlsson, E.A.: Software reuse: a holistic approach. John Wiley & Sons, Inc., (1995)

Kawamoto, K., Lobach, D.F.: Proposal for Fulfilling Strategic Objectives of the U.S. Roadmap for National Action on Decision Support through a Service-oriented Architecture Leveraging HL7 Services. Journal of the American Medical Informatics Association **14**(2), 146-155 (2007)

Khan, M.U.: Architectural Constraints in the Model-Driven Development of Self-Adaptive Applications. In: Roland, R., Kurt, G. (eds.), vol. 9. (2008)

Ko, A.J., DeLine, R., Venolia, G.: Information Needs in Collocated Software Development Teams. Paper presented at the Proceedings of the 29th international conference on Software Engineering,

Krogstie, J.: Model-Based Development and Evolution of Information Systems: A quality approach, 2012 ed. Springer, (2012)

Krueger, C.W.: Software reuse. ACM Computing Surveys (CSUR) **24**(2), 131-183 (1992)

Kuhn, K.A., Lenz, R., Elstner, T., Siegele, H., Moll, R.: Experiences with a generator tool for building clinical application modules. Methods of information in medicine **42**(1), 37-44 (2003)

Kuzniarz, L., Staron, M., Wohlin, C.: An empirical study on using stereotypes to improve understanding of UML models. In: 2004, pp. 14-23. IEEE

Kärnä, J., Tolvanen, J.P., Kelly, S.: Evaluating The Use of Domain-Specific Modeling in Practice. (2009)

Lago, P., Muccini, H., van Vliet, H.: A scoped approach to traceability management. Journal of Systems and Software **82**(1), 168-182 (2009)

Lenz, R., Beyer, M., Kuhn, K.A.: Semantic integration in healthcare networks. International journal of medical informatics **76**(2-3), 201-207 (2007)

Lenz, R., Kuhn, K.A.: Towards a continuous evolution and adaptation of information systems in healthcare. International journal of medical informatics **73**(1), 75-89 (2004)

Limón, A.E., Garbajosa, J.: The Need for a Unifying Traceability Scheme. In: European Conference on Model Driven Architecture - Traceability Workshop 2005, Nuremberg, 2005, November 8 2005

Luis M. Camarinha-Matos, Rosas, J.o., Ferrada, F., Oliveira, A.I.s., Afsarmanesh, H., Brielmann, M.: ICT & Ageing Scenarios. In. BRAID Project, (2011)

Lundell, B., Lings, B.: Changing perceptions of CASE technology. The Journal of Systems & Software **72**(2), 271-280 (2004)

MacDonald, A., Russell, D., Atchison, B.: Model-driven development within a legacy system: an industry experience report. Software Engineering Conference, 2005. Proceedings. 2005 Australian, 14-22 (2005)

Magnusson, L., Hanson, E., Borg, M.: A literature review study of information and communication technology as a support for frail older people living at home and their family carers. Technology and Disability **16**(4), 223-235 (2004)

Mattsson, A., Lundell, B., Lings, B., Fitzgerald, B.: Experiences from representing software architecture in a large industrial project using model driven development. In: Sharing and Reusing Architectural Knowledge-Architecture, Rationale, and Design Intent, 2007. SHARK/ADI'07: ICSE Workshops 2007. Second Workshop on 2007, pp. 6-6. IEEE

Mellor, S.J.: MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley Professional, (2004)

Miles, M.B., Huberman, A.M.: Qualitative data analysis : an expanded sourcebook, 2nd ed. Sage Publications, Thousand Oaks (1994)

Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. In: Miller, J., Mukerji, J. (eds.). pp. 1-62. Object Management Group (OMG), (2003)

Mohagheghi, P., Dehlen, V.: Where Is the Proof?-A Review of Experiences from Applying MDE in Industry. Model-Drivern Architecture-Foundations and Applications: 4th European Conference, Ecmda-Fa 2008, Berlin, Germany, June 9-13, 2008, Proceedings (2008)

Mohagheghi, P., Dehlen, V., Neple, T.: Definitions and approaches to model quality in model-based software development-A review of literature. Information and Software Technology **51**(12), 1646-1669 (2009a)

Mohagheghi, P., Fernandez, M., Martell, J., Fritzsche, M., Gilani, W.: MDE adoption in industry: Challenges and success criteria. Models in Software Engineering, 54-59 (2009b)

MPOWER Consortium: FREE MPOWER. http://sourceforge.net/projects/free-mpower (2008a). Accessed November 9 2012

MPOWER Consortium: Middleware platform for eMPOWERing cognitive disabled and elderly. http://www.sintef.no/mpower (2008b). Accessed November 9 2012

Nasjonal IKT: Tjenesteorientert arkitektur i spesialisthelsetjenesten. In, vol. 2008. vol. November 9. (2011)

Naur, P., Randell, B.: Software Engineering: Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 7th to 11th October 1968. Scientific Affairs Division, (1968)

Norwegian Ministry of Social Affairs, Norwegian Ministry of Health: Te@mwork 2007 - Electronic Interaction in the Health and Social Sector. In., p. 25. Directorate for Health and Social Affairs, (2004)

OASIS Open: Reference Model for Service Oriented Architecture 1.0. In: C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz (eds.). OASIS Open, (2006)

Oates, B.J.: Researching information systems and computing. Sage Publications Ltd, (2006)

Object Management Group: Service oriented architecture Modeling Language (SoaML). In: "In Process" Version of SoaML: 1.0 Beta 2. OMG, (2009)

Object Management Group (OMG). http://www.omg.org/. Accessed November 9 2012

Object Management Group (OMG): MDA Guide Version 1.0.1. In: Miller, J., Mukerji, J. (eds.). pp. 1-62. Object Management Group, (2003)

Object Management Group (OMG): UML 2.0 Superstructure FTF Rose model containing the UML 2 metamodel. In., vol. 2006-08-31. Object Management Group (OMG), (2005)

Object Management Group (OMG): Object Constraint Language (OCL), Version 2.0. In., pp. 1-232. Object Management Group, (2006)

Object Management Group (OMG): UML 2.1.2 Superstructure and Infrastructure. In., vol. formal/2007-11-04. Object Management Group (OMG), (2007)

Omar, W.M.: E-health support services based on service-oriented architecture. IT professional **8**(2), 35 (2006)

Park, J.: Information systems interoperability: What lies beneath? ACM transactions on information systems **22**(4), 595 (2004)

Parviainen, P., Takalo, J., Teppola, S., Tihinen, M.: Model-Driven Development. (2009)

Purchase, H.C., Colpoys, L., McGill, M., Carrington, D.: UML collaboration diagram syntax: an empirical study of comprehension. In: 2002, pp. 13-22. IEEE

Raghupathi, W., Kesh, S.: Interoperable electronic health records design: towards a service-oriented architecture. e-Service Journal **5**(3), 39-57 (2008)

Raistrick, C.: Applying MDA and UML in the Development of a Healthcare System. In: UML Modeling Languages and Applications, vol. Volume 3297/2005. Lecture Notes in Computer Science, pp. 203-218. Springer Berlin / Heidelberg, (2005)

Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments. Software Engineering, IEEE Transactions on **36**(1), 96-118 (2010)

Rine, D.C., Nada, N.: An empirical study of a software reuse reference model. Information and Software Technology **42**(1), 47-65 (2000)

Rosen, M.: MDA, SOA, and Technology Convergence. In: David S. Frankel, John Parodi (eds.) The MDA Journal Straight from the Masters. pp. 62-79. Meghan-Kiffer Press, Tampa, Florida, USA (2004)

Rubin, K.S., Beale, T., Blobel, B.: Modeling for Health Care. In: Person-Centered Health Records. Health Informatics, pp. 125-146. Springer New York, New York (2005)

Sametinger, J.: Software engineering with reusable components. Springer Verlag, (1997)

Schulte, M.: Model-based integration of reusable component-based avionics systems - a case study. In: Object-Oriented Real-Time Distributed Computing, 2005.

ISORC 2005. Eighth IEEE International Symposium on, 18-20 May 2005 2005, pp. 62-71

Seaman, C.B.: Qualitative Methods. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering. pp. 35-62. Springer London, (2008)

Selic, B.: A Systematic Approach to Domain-Specific Language Design Using UML. 10th IEEE ISORC **7** (2007)

Shaikh, A., Wiil, U.K., Memon, N.: Evaluation of tools and slicing techniques for efficient verification of UML/OCL class diagrams. Advances in Software Engineering **2011**, 5 (2011)

Sharma, S., Rai, A.: CASE deployment in IS organizations. Communications of the ACM **43**(1), 80-88 (2000)

Shull, F., Singer, J., Sjøberg, D.I.K.: Guide to Advanced Empirical Software Engineering. Springer Verlag, London (2007)

Sim, S.E., Lethbridge, T.C.: Software Engineering Data Collection for Field Studies. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering. pp. 9-34. Springer London, (2008)

Spinellis, D., Szyperski, C.: How is open source affecting software development? IEEE software **21**(1), 28-33 (2004)

Stahl, T., Völter, M.: Model-driven software development: technology, engineering, management. Wiley, Chichester (2006)

Standish Group International: The Chaos Report. In. Standish Group International, (1994)

Staron, M.: Improving modeling with UML by stereotype-based language customization. Blekinge Institute of Technology (2005)

Staron, M.: Adopting Model Driven Software Development in Industry–A Case Study at Two Companies. Proceedings of the MoDELS/UML conference (2006)

Stead, W.W.: Integration and Beyond. Journal of the American Medical Informatics Association **7**, 135 (2000)

Tattersall, R.: The expert patient: a new approach to chronic disease management for the twenty-first century. Clin Med **2**(3), 227-229 (2002)

The Middleware Company: Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach. In: MDA Productivity Case Study. (2003)

The MPOWER Consoritum: MPOWER Website. http://www.mpower-project.eu (2007). Accessed June 15 2007

Tolvanen, J.P., Rossi, M.: MetaEdit+: defining and using domain-specific modeling languages and code generators. In: 2003, pp. 92-93. ACM

Trask, B., Paniscotti, D., Roman, A., Bhanot, V.: Using model-driven engineering to complement software product line engineering in developing software defined radio components and applications. In: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications 2006, pp. 846-853. ACM

U.S. Department Of Health and Human Services, Food and Drug Administration Center for Devices, Radiological Health Center for Biologics Evaluation and Research: General Principles of Software Validation; Final Guidance for Industry and FDA Staff. (2002).

Van Someren, M.W., Barnard, Y.F., Sandberg, J.A.C.: The think aloud method: A practical guide to modelling cognitive processes. Academic Press London, (1994)

Venkatesh, V., Davis, F.D.: A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. Management Science **46**(2), 186-204 (2000)

Walderhaug, S.: Evaluation of a Model-Driven Development Toolchain for Healthcare. Automated Software Engineering Journal **Revision submitted on September 1, 2012** (2012)

Walderhaug, S., Mikalsen, M., Benc, I., Loniewski, G., Stav, E.: Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project. In: Bailey, T. (ed.) From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture, Berlin, Germany 2008a. European Software Institiute

Walderhaug, S., Mikalsen, M., Salvi, D., Svagård, I., Ausen, D., Kofod-Petersen, A.: Towards Quality Assurance of AAL Services. In: Blobel, B. (ed.) Phealth 2012: Proceedings of the 9th International Conference on Wearable Micro and Nano Technologies for Personalized Health 2012, p. 296. IOS Press

Walderhaug, S., Stav, E., Johansen, U., Olsen, G.K.: Traceability in Model-driven Software Development. In: Tiako, P.F. (ed.) Designing Software-Intensive Systems - Methods and Principles. pp. 133-160. IGI Global, Information Science Reference, Hersey, PA (2008b)

Walderhaug, S., Stav, E., Mikalsen, M.: Reusing models of actors and services in smart homecare to improve sustainability. Stud Health Technol Inform **136**, 107-112 (2008c)

Walker, J., Pan, E., Johnston, D., Adler-Milstein, J., Bates W., D., Middleton, B.: The Value of Health Care Information Exchange and Interoperability. Health Tracking **5**(10) (2005)

Wancata, J., Musalek, M., Alexandrowicz, R., Krautgartner, M.: Number of dementia sufferers in Europe between the years 2000 and 2050. European Psychiatry **18**(6), 306-313 (2003)

Warmer, J.B.: "Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition". (2003)

Winkler, S., Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Software & Systems Modeling **9**, 529-565 (2009). doi:10.1007/s10270-009-0145-0

Winnem, O.M., Walderhaug, S.: Distributed, role based, guideline based decision support. In: E-he@lth in Common Europe 2002, pp. 101-109. Springer

World Wide Web Consortium (W3C): Web Services Architecture. In: Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D. (eds.). W3C, (2004)

World Wide Web Consortium (W3C): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. In: Chinnic, R., Jean-Jacques Moreau, Ryman, A., Weerawarana, S. (eds.). W3C, (2007)

Part II: Collection of papers

Part II Collection of papers

P1: **Walderhaug, S**., Mikalsen, M., Hartvigsen, G., Stav, E., Aagedal, J.: *Improving systems interoperability with model-driven software development for healthcare*. Stud Health Technol Inform 129(Pt 1), 122-126 (2007)

P2: **Walderhaug, S**., Stav, E., Mikalsen, M.: *The MPOWER Tool Chain - Enabling Rapid Development of Standards-based and Interoperable HomeCare Applications*. In: Sandnes, F.E. (ed.) Norsk Informatikk Konferanse (NIK), Oslo, October 2007 2007, pp. 103-107. TAPIR (2007)

P3: **Walderhaug, S.,** Mikalsen, M., Benc, I., Loniewski, G., Stav, E.: *Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project*. In: Bailey, T. (ed.) From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture, Berlin, Germany 2008. European Software Institute

P4: **Walderhaug, S.,** Stav, E.: *Experiences from model-driven development of homecare services: UML profiles and domain models*. Paper presented at the 2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems (MOTHIS 2008) in Toulouse, France. *The paper was selected as the session's best paper*.

P5: **Walderhaug, S**., Stav, E., Johansen, U., Olsen, G.K.: *Traceability in Model-driven Software Development*. In: Tiako, P.F. (ed.) Designing Software-Intensive Systems - Methods and Principles. pp. 133-160. IGI Global, Information Science Reference, Hersey, PA (2008)

P6: **Walderhaug, S.,** Stav, E., Mikalsen, M.: *Reusing models of actors and services in smart homecare to improve sustainability*. Stud Health Technol Inform 136, 107-112 (2008)

P7: Holthe, T., **Walderhaug, S**.: *Older people with and without dementia participating in the development of an individual plan with digital calendar and message board*. Journal of Assistive Technologies 4(2), 15-25 (2010)

P8: **Walderhaug, S.,** Hartvigsen, G., Stav, E.: *Model-driven traceability in healthcare information systems development*. Stud Health Technol Inform 160(Pt 1), 242-246 (2010)

P9: Stav, E., **Walderhaug, S**., Mikalsen, M., Hanke, S., Benc, I.: Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned. International Journal of Medical Informatics (2011). doi:10.1016/j.ijmedinf.2011.03.007

P10: **Walderhaug, S.:** *Evaluation of a Model-Driven Development Toolchain for Healthcare*. Automated Software Engineering Journal, revision submitted on September 1st, 2012

Related papers not included in this thesis:

1. **Walderhaug, S**., Stav, E., Johansen, U., Aagedal, J., 2006. *Towards a Generic Solution for Traceability in MDD*, in: Aagedal, J., Neple, T., Oldevik, J. (Eds.), European Concerence on Model Driven Architecture Traceability Workshop (ECMDA-TW). SINTEF, Bilbao, Spain.
2. Droes, R.-M., Mulvenna, M., Nugent, C., Finlay, D., Donnelly, M., Mikalsen, M., **Walderhaug, S**., Kasteren, T.v., Krose, B., Puglia, S., Scanu, F., Migliori, M.O., Ucar, E., Atlig, C., Kilicaslan, Y., Ucar, O., Hou, J., 2007. *Healthcare Systems and Other Applications*, in: Maurice, M. (Ed.), Pervasive Computing, IEEE, pp. 59-63.
3. Mikalsen, M., **Walderhaug, S**., Meland, P.H., Winnem, O.M., 2007. *Linkcare -enabling continuity of care for the chronically ill across levels and profession*. Stud Health Technol Inform 129, 3-7.
4. Pitsillides, A., Themistokleous, E., Samaras, G., **Walderhaug, S**., Winnem, O.M., 2007. *Overview of MPOWER: Middleware Platform for the Cognitively Impaired and Elderly*, IST Africa 2007, Maputo, Mosambique.
5. Prazak, B., Hochgatterer, A., Holthe, T., **Walderhaug, S**., 2007. *User Requirements as Crucial Determinants for the Development of New Technological Solutions for Elderly Care - Exemplified in an European Project*, in: G. Eizmendi, Azkoitia, J.M., Craddock, G.M. (Eds.), AAATE. IOSPress, San Sebastian, Spain.
6. Loniewski, G., Ramon, E.L., **Walderhaug, S.**, Franco, S.M., Esteve, J.J.C., Marco, E.S., 2008. *Data Management in an Intelligent Environment for Cognitive Disabled and Elderly People*, EHealth 2008. Springer London, UK.
7. Mikalsen, M., Hanke, S., Fuxreiter, T., **Walderhaug, S**., Wienhofen, L., 2009. *Interoperability services in the MPOWER Ambient Assisted Living platform*. Stud Health Technol Inform 150, 366-370.
8. Mikalsen, M., **Walderhaug, S**., Salvi, D., Hanssen, G.K., 2012. *Key Technological Success Features for a Domain Specific Open Software Ecosystem for Ambient Assisted Living*, in: Eichler, G., Wienhofen, L.W.M., Kofod-Petersen, A., Unger, H. (Eds.), 12th International Conference on Innovative Internet Community Systems. Springer Verlag, Trondheim, Norway.
9. **Walderhaug, S**., Mikalsen, M., Salvi, D., Svagård, I., Ausen, D., Kofod-Petersen, A., 2012. *Towards Quality Assurance of AAL Services*, in: Blobel, B. (Ed.), Phealth 2012: Proceedings of the 9th International Conference on Wearable Micro and Nano Technologies for Personalized Health. IOS Press, p. 296.

Paper 1

Walderhaug, S., Mikalsen, M., Hartvigsen, G., Stav, E., Aagedal, J.: *Improving systems interoperability with model-driven software development for healthcare*. Stud Health Technol Inform 129(Pt 1), 122-126 (2007)

Paper 2

Walderhaug, S., Stav, E., Mikalsen, M.: *The MPOWER Tool Chain - Enabling Rapid Development of Standards-based and Interoperable HomeCare Applications*. In: Sandnes, F.E. (ed.) Norsk Informatikk Konferanse (NIK), Oslo, October 2007 2007, pp. 103-107. TAPIR (2007)

Paper 3

Walderhaug, S., Mikalsen, M., Benc, I., Loniewski, G., Stav, E.: *Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project*. In: Bailey, T. (ed.) From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture, Berlin, Germany 2008. European Software Institute

Paper 4

Walderhaug, S., Stav, E.: *Experiences from model-driven development of homecare services: UML profiles and domain models*. Paper presented at the 2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems (MOTHIS 2008) in Toulouse, France. *The paper was selected as the session's best paper*.

Paper 5

Walderhaug, S., Stav, E., Johansen, U., Olsen, G.K.: *Traceability in Model-driven Software Development*. In: Tiako, P.F. (ed.) Designing Software-Intensive Systems - Methods and Principles. pp. 133-160. IGI Global, Information Science Reference, Hersey, PA (2008)

Paper 6

Walderhaug, S., Stav, E., Mikalsen, M.: *Reusing models of actors and services in smart homecare to improve sustainability*. Stud Health Technol Inform 136, 107-112 (2008)

Paper 7

Holthe, T., **Walderhaug, S**.: *Older people with and without dementia participating in the development of an individual plan with digital calendar and message board*. Journal of Assistive Technologies 4(2), 15-25 (2010)

Paper 8

Walderhaug, S., Hartvigsen, G., Stav, E.: *Model-driven traceability in healthcare information systems development.* Stud Health Technol Inform 160(Pt 1), 242-246 (2010)

Paper 9

Stav, E., **Walderhaug, S**., Mikalsen, M., Hanke, S., Benc, I.: Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned. International Journal of Medical Informatics (2011). doi:10.1016/j.ijmedinf.2011.03.007

Paper 10

Walderhaug, S.: *Evaluation of a Model-Driven Development Toolchain for Healthcare*. Automated Software Engineering Journal, revision submitted on September 1st, 2012

Appendix A. Interview Guide (semi-structured)

- Please tell me about your education with focus on software engineering courses, programming languages and other relevant course
- What do you think about the exercise?
- Did you understand the exercise scenario
    - Was the difficulty ok?
- What do you thing about the first part of the exercise: the computation independent modelling
    - Use case modelling
        - Did you use the actors library
        - Did you find the actors you were looking for
    - Feature modelling
    - Trace link modelling
- What do you think about the second part: platform specific modelling
    - Information modelling
        - Which background information did you use to create the information model
        - Did you get all the details necessary?
    - Service modelling
        - How did you identify services
        - How did you identify interface operations
        - Service model template
        - Which information did you use for message design
        - Did you use the traceability information
- What did you think about the final part: platform specific modeling
    - Generating RTF and HTML documentation
        - What do you think was the most useful?
    - Using HTML documentation for traceability
    - DDL Transformation
    - WSDL Transformation
- Using the DDL and WSDL files for database and
    - General comments?
- Would you like to learn more about MDD

    - Take more courses at the university?
- Any other comments?

Appendix B. Experiment Scenario description

MedList – Shared medication list

A common problem for patients, and especially elderly, is the management of medication. The problems include:

1) *Read information about dosage and description about when and how the medication should be taken*
2) *Manage updated medication: change in dosage and amount*
3) *Share problems with medication between the patient's family, visiting nurses and the GP.*
4) *Report back to GP if medication has been taken or not.*

In Ulvilla, a nice village just outside of Verdal in Nord-Trøndelag, the happy couple Odd and Anna lives in a small house on their farm Elnes Nedre. They retired seven years ago, and now their oldest son Harald is running the farm together with his wife Åse.

Odd had a stroke 5 years ago, and has since then been on medication. Just recently he got a new type of medicine from his GP, Dr Abbas. Odd and Anna went to see Dr. Abbas together, but none of them really understood what he said, and they were afraid to ask – after all he is a respected doctor in Verdal. As a result of this uncertainty, Odd takes the medication too often – "to be on the safe side"

After one week of "misuse", Odd feels sick and need to see the doctor. It is soon revealed that he as taken to much medicine. Harald and Åse are really angry, and after some phone calls, a new research project is started: MedList. This project will address the problems 1-4 in the list above.

The project will:

a) *Build upon a SOA platform, using web services in the local health network*
b) *Have an easy to use client installed at the GP, patient and the patient's relatives*

c) Share information about medication between the defined actors.

Appendix C. Professional Survey Questionnaire

The questionnaire (in Norwegian) is available from the author on request.

Appendix D. Interview analysis

Raw material and analysis results are available on request to the author.

ISBN xxx-xx-xxxx-xxx-x