

UNIVERSITY OF OSLO
Department of informatics

PROSERVE

**Process-driven Business Service Identification
with Context Support**

*Metamodel and Notation with Tool Support through
Hierarchical Representations in Marama MetaTools
and Transformations in MOFScript*

Master thesis

60 credits

Espen Møller

1st of November 2008



Abstract

Services need to be represented at both a business level and a software level in order to separate concerns in the different contexts. However, the notion of service in business is not well aligned with the notion of service used in service-oriented architecture (SOA). Moreover, the current method for describing business in a SOA context is not focused on business services, but on business processes. State-of-the-art modelling tools for SOA need to resolve how to rapidly identify business services from business processes, support technology independent representation of these and relate the subset of IT outsourced services to a software service context.

PROSERVE is a metamodel and tool that separates contexts for services in business and software. It provides proof-of-concepts for separation of concerns between these domains and a solution for automated identification of business services from business process models.

This thesis have four main contributions: A service context scheme was created to provide (i) rules for abstracting different views upon a common model (context-views) and (ii) rules for how services can interact (context-interaction); creation of mapping rules from a business process metamodel enabled model-to-text transformation execution to (iii) identify business service artefacts automatically; a service tree was created as a notational artefact for the metamodel in order to provide (iv) selective abstractions (context-overview). PROSERVE suggests a resolution for how to use the service metaphor without considering underlying technology specifications of them while dependencies between the domains are traced. The contributions made in this thesis were intended to form a technical foundation for future improvement of decision-support for business people on outsourcing of services to IT with coupling to SOA design models.

Acknowledgements

I would like to express my special thanks to my supervisors Dr. Arne-Jørgen Berre (chief scientist, SINTEF ICT), Prof. Dr. John Hosking (academic director, University of Auckland) and Prof. Dr. John Grundy (head of department, University of Auckland) who have given me an opportunity to learn about state-of-the-art research in software engineering and given my work direction both when the path of research was bright and when it was dark.

Their dedication to their work and vast technological overview has inspired me beyond my own expectations and introduced me to a whole new level of knowledge and learning.

Through my affiliation with the University of Auckland where I stayed as a visitor during broad parts of the thesis work, I had access to a network of scientist, engineers and fellow students working on related research topics. Among these I would especially like to thank:

- Richard Li (PhD student and lecturer, University of Auckland) for sharing his knowledge on automatic diagram layout and hierarchical representations through his work on Enterprise Modelling Language.
- Jun Huh (software engineer, University of Auckland) for providing support and help on Marama. In particular, his contribution on implementing a simple tree structure was crucial for providing proof-of-concept for the service tree.
- Christian Hirsch (PhD student, University of Auckland), Rainbow Cai (PhD student, University of Auckland) and Pei-Shan Yap (Post-graduation student, University of Auckland) for contributing in important discussions on the work conducted in this thesis.

Through my affiliation with SINTEF I had access to a similar research network. Among these I would especially like to thank:

- Gørán K. Olsen (research scientist, SINTEF ICT) for providing help on the MOFScript tool.
- Erlend Stav (senior research scientist, SINTEF ICT) for feedback on my work.

For insight and discussion on health care matters and architecture I would like to thank Andreas Brunvoll (IT-architect at Ullevål University Hospital) and Øyvind Aassve (information architect, Rikshospitalet University Hospital). Participation in the SOA-in-practice network has also given me valuable input to my work.

I would also like to thank Stine Holm (SINTEF) and Robyn Young (University of Auckland) for practical help.

Finally, I would like to thank my friends for being there when I was frustrated, exhausted and lost. I would especially like to thank my friends Ann Blomberg for proof-reading and Anne-Laure Tougeron for countless coffee chats on services.

This thesis is written in New Zealand English. It is **recommended that it is printed in colours**, since colour codes are used.

Table of Contents

Abstract.....	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
List of Tables	x
Chapter 1. Introduction.....	1
1.1. Background and Motivation.....	1
1.2. Objective and Scope	1
1.3. Research Method	1
1.4. Outline of the Thesis.....	3
Chapter 2. Background	6
2.1. Service Science.....	6
2.2. Model-Driven Architecture	6
2.2.1. Metamodels.....	7
2.2.2. Models and Diagrams.....	8
2.2.3. Model Transformation.....	8
2.3. Enterprise Architecture	9
2.4. Business Process Modelling.....	11
2.4.1. Business Process Modeling Notation (BPMN).....	12
2.5. Service-Oriented Architecture (SOA).....	12
2.5.1. OASIS SOA Reference Model (OSOAR).....	13
2.5.2. UML Profile and Metamodel for Services (UPMS).....	13
2.5.3. SOMA	14
2.6. Summary	14
Chapter 3. Problem Analysis	15
3.1. A Case Study: Health Care Appointment Reminders for Patients.....	15
3.2. Literature Review	19
3.2.1. Alignment of SOA Services, Customer-oriented Services and E-services.....	19
3.2.2. Granularity in Services and Business Processes	22
3.2.3. Service Interaction	23
3.2.4. From Business Processes to Software Services	24
3.2.5. Scalability of Services in Conceptual Modelling.....	25
3.3. Research Needs.....	25
3.4. Hypothesis.....	26
3.5. Success Criteria	27
3.6. Approach.....	27
3.7. Summary	29
Chapter 4. Requirements	30
4.1. Service Context Scheme Requirements	30
4.2. Tool Requirements.....	30
4.3. Evaluation of State-of-the-art Tool Frameworks and Methodologies for Services Modelling .	32
4.4. Summary	35

Chapter 5. Theoretical Solution	36
5.1. PROSERVE Metamodel	36
5.1.1. Taxonomy	36
5.1.2. Core	37
5.1.3. Context Scheme	38
5.1.4. Service Splitting	41
5.1.5. Views	42
5.1.6. Metamodel	45
5.1.7. Limitations	48
5.2. Process-driven Business Service Identification	48
5.2.1. Behaviour	51
5.3. Service Architecture	51
5.4. Summary	52
Chapter 6. The PROSERVE Tool	53
6.1. Subset of Problem to be Proven	53
6.2. Tools and Technologies	54
6.2.1. Meta-tools	54
6.2.2. Model Transformation Framework	55
6.3. Notation	56
6.3.1. Shapes	56
6.3.2. Trees	57
6.4. Diagrams	57
6.4.1. Business View	57
6.4.2. Integrated View	58
6.4.3. Software View	58
6.4.4. All views	59
6.4.5. Diagram Behaviour	59
6.5. System Design	61
6.5.1. Components	61
6.5.2. User Event Handlers	63
6.5.3. BPMN Modelling Tool	65
6.5.4. Model Transformer	65
6.6. Realization Overview	66
6.6.1. Transformer Integration	67
6.6.2. Tree Algorithm	67
6.6.3. Model Transformation	68
6.7. Summary	69
Chapter 7. Design of Experiment, Tests and Analysis	70
7.1. Appointment Process for Out-patients at a Hospital	70
7.2. Tests for Service Context Scheme	74
7.3. Tests for PROSERVE Tool	75
7.4. Application on the Case Study for Out-patients at a Hospital	76
7.5. Cognitive Dimensions Analysis of the Service Tree Structure	80
7.6. Summary	83
Chapter 8. Evaluation and Discussion	84
8.1. Evaluation of Service Context Scheme Requirements	84
8.2. Evaluation of Tool Requirements	85
8.3. Cognitive Dimensions Evaluation	86

8.4. Comparison with State-Of-The-Art Solutions.....	87
8.5. Evaluation of Success Criteria and Hypothesis	87
8.5.1. Success Criteria.....	87
8.5.2. Hypothesis	89
8.6. Discussion and Critique	89
8.6.1. PROSERVE Alignment with Existing Service Identification Approaches	89
8.6.2. PROSERVE Alignment with Enterprise Architecture and Service-oriented Architecture	91
8.6.3. PROSERVE Alignment with Software Development Process	92
8.6.4. PROSERVE weakness	92
8.6.5. Gained Knowledge.....	93
8.7. Summary	95
Chapter 9. Conclusion and Future Work	96
9.1. Conclusion.....	96
9.2. Claimed Contributions	96
9.3. Future Work	97
9.3.1. Completion of Current Work.....	97
9.3.2. A Health Care Challenge.....	98
9.3.3. Improved Technology Artefacts for Service Interoperability.....	100
9.3.3.1. BIZSERVE	101
9.3.3.2. INTSERVE	103
9.3.3.3. SOFTSERVE	103
9.3.4. SERVE for Ullevål University Hospital.....	104
References.....	107
Appendix I: LIST OF ABBREVIATIONS	112
Appendix II: PROSERVE TOOL INTERFACE	113
Business View.....	113
Integration View.....	115
Software View.....	116
All Views	117
User Event Handlers.....	119
Appendix III: PROSERVE DIAGRAMS.....	120
Appendix IV: REFERENCE TO DELIVERED TOOL ARTEFACT	122

List of Figures

Figure 1: Contributions	1
Figure 2: Use of Research Method in the Thesis Work.....	3
Figure 3: Structure of Thesis.....	4
Figure 4: Concept of Layered Metamodel Hierarchy. Adapted from [13].	8
Figure 5: Mapping of Concepts between Different Levels of Abstraction. Adapted from [15].....	9
Figure 6: TOGAF Architecture Development Cycle [18].....	10
Figure 7: Send Sub-Process	16
Figure 8: ARSP Main Process.....	17
Figure 9: Exception Sub-Process	17
Figure 10: Departmental-Follow-Up Sub-Process	17
Figure 11: Illustration of Service Delivery.	18
Figure 12: Service Interaction [33].	20
Figure 13: Service-Oriented Business. Adapted from [36].....	23
Figure 14: Service Interaction. Adapted from [44].	24
Figure 15: PROSERVE Concepts.....	28
Figure 16: Context of Work.....	29
Figure 17: COMET Framework for Business Modelling. Adapted from [53].	32
Figure 18: EML Process Overlay [49].	33
Figure 19: SOMA Service Identification. Adapted from [19].	34
Figure 20: Resolved Concepts in the Metamodel.	36
Figure 21: Service Contract Agreement.....	38
Figure 22: Service Context Scheme.	40
Figure 23: Service Network.....	41
Figure 24: Service Splitting.....	42
Figure 25: Use Case of PROSERVE.....	43
Figure 26: Views on the PROSERVE Model.....	44
Figure 27: Service Delegation.....	45
Figure 28: PROSERVE Metamodel.	48
Figure 29: Resolved Concept.	49
Figure 30: Derivation of PROSERVE Artefacts from BPMN.	51
Figure 31: Service Architecture.....	52
Figure 32: Resolved Concept.	53
Figure 33: Marama Architecture [54].	55
Figure 34: MOFScript Tool Architecture [66].	56
Figure 35: Diagrams in Business View.....	58
Figure 36: Diagram in Integrated View.....	58
Figure 37: Diagram in Software View.	59
Figure 38: Domain independent diagrams.	59
Figure 39: Result of Shape Intersection between EnterpriseSoftware and Service.....	60
Figure 40: Adding Requisition.	60
Figure 41: Tree Scalability.....	60
Figure 42: Service Contract Agreement.....	61
Figure 43: Main Components for PROSERVE Tool.....	62

Figure 44: Tree Building on Import.....	63
Figure 45: Importing Instances from Model.	64
Figure 46: BPMN to Service Transformation.	66
Figure 47: Shape Definitions in Marama.	66
Figure 48: Linking Shapes to Entities.....	67
Figure 49: Tree Metamodel. Adapted from [69].	67
Figure 50: Simple Tree.	67
Figure 51: Inheritance from Control is Necessary to Make an Entity a Tree Node.	68
Figure 52: Part of Rules for Deriving Behaviour.	69
Figure 53: Appointment Process for Out-patients.	70
Figure 54: BookAppointment Sub-Process.	71
Figure 55: AppointmentReminder Sub-Process.	71
Figure 56: SendAppointmentReminder.	72
Figure 57: ReminderException Sub-Process.....	72
Figure 58: DepartmentalFollowUp Sub-Process.	73
Figure 59: ProcessAppointment Sub-Process.	73
Figure 60: PatientRegistration Sub-Process.....	73
Figure 61: HematologyAndRadiology Sub-Process.	74
Figure 62: MedicalExamination Sub-Process.....	74
Figure 63: Generated Services.	74
Figure 64: Automatic Visualization of derived Services in Business Service Contract Diagrams.....	76
Figure 65: Selective Abstraction.	77
Figure 66: Adding Context through Ownerships.....	77
Figure 67: Outsourcing Service to IT.	78
Figure 68: Delegating to Software Services in Integration View.	78
Figure 69: Tracing Superior Services.	79
Figure 70: Tracing Inferior Services.	79
Figure 71: Service Behaviour of BookAppointment Service.	80
Figure 72: Scalable Information.	82
Figure 73: Juxtaposibility.	83
Figure 74: Trade-Offs - Determining the Degree of Automation.	101
Figure 75: Vertical Mappings.	102

List of Tables

Table 1: BPMN Entities	12
Table 2: Concerns of Services in Business and Software Domains.....	22
Table 3: Research Needs.....	26
Table 4: State-of-the-Art Evaluation.....	35
Table 5: Abstraction of Views Based on Participant Types.	43
Table 6: Evaluation of Features.....	54
Table 7: Entity Shapes (Icons).	57
Table 8: Event Handlers.	67
Table 9: Tree-Node Properties.....	68
Table 10: Cognitive Dimensions Evaluation.....	86
Table 11: Tool Comparison.	87

Chapter 1. Introduction

This chapter outlines an overview of the thesis work and starts by introducing context and motivation. Then objectives, scope and goals for contributions are described. Finally, the research method is presented and aligned with the structure of the thesis.

1.1. Background and Motivation

Worldwide, services represent a growing segment of the developed and the developing economies [1]. Technology is one key driving force in this development. This has lead to a need for companies to understand how technology and business practices can intersect to get the most out of each. It is therefore identified a demand for overlapping skills and knowledge in the juncture between business and IT in the service industry. In order to capture this knowledge and to manage services more efficiently, new models, methodologies, processes and software tools have to be created.

Business modelling or enterprise architecture is an approach for understanding a business. The emergence of service-oriented architecture (SOA) as a business-driven philosophy has created a need to couple software development models and business models. In SOA, the service metaphor has traditionally been limited to denote software services.

However, the shift towards service orientation in economies creates an increased need for representing services independently of technology as a means of making grounds for decision-making and describing what the business is about. A service metaphor is a means for assigning responsibilities both to human entities and application entities in the organization. State-of-the-art modelling tools and methodologies tools lack support for representing different types of service representations, separating concerns and dealing with the context they work in.

Business people lack modelling support to describe a service-oriented business. In enterprise architecture, business process modelling is usually used to describe the core of the business. However, it can be argued that service modelling may provide an equally important perspective on what the business is about. Process-oriented service identification may be a mechanism for bridging the gap between these perspectives.

1.2. Objective and Scope

The main objective of this thesis is to provide improved modelling artefacts for the business domain as a response to the need to capture service descriptions between business and IT. A future objective can be to create improved decision support for outsourcing of services to IT. However, this thesis objective is to resolve a technical foundation for this through separation of concerns and automated process-driven identification of business services. It is aimed at providing contributions for:

- Context-views: Separation of contexts for service metaphors.
- Context-interaction: Rules for how services can interact in a heterogeneous environment.
- Context-overview: Improved overview in diagrams by abstracting details.
- Service identification: Identification of business services from business process models.

The contributions are depicted in **Figure 1**. An explanation of the concepts is given in section [3.3](#).

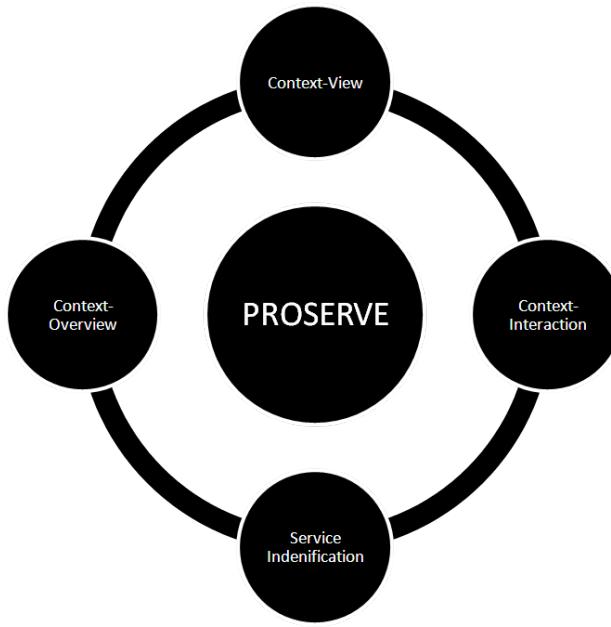


Figure 1: Contributions.

A case study from health care is used to exemplify needs for research and to evaluate the solution. The work is aligned with a higher level discussion on service-orientation of health enterprises. Finally, based on this thesis, future work is proposed to find improved technologies for supporting service-orientation of organizations and semantic service interoperability by considering challenges from health care.

The context of the research is a merge of three conceptual areas:

- The use of service as a metaphor in business and software development.
- The use of conceptual modelling and model-driven techniques to represent services and to identify them.
- Visualization of service artefacts in diagrams.

To scope down the size of the problem area to be analyzed, only the following areas are considered:

- Service Science (different views on services).
- Service-Oriented Architecture including E-services.
- Model-driven Architecture.
- Service artefact visualization.

1.3. Research Method

The main goal in technology research is to create new artefacts or to improve existing artefacts in order to support identified needs [2]. Identified needs are described in section [3.3](#). Requirements are then specified in Chapter [4](#) in order to evaluate this thesis' proposed solution.

In section 3.4, the hypothesis of this thesis is stated, and a set of predictions are derived in order to test this hypothesis. The predictions are defined in a way that makes them possible to falsify. They will serve as success criteria for the developed artefacts.

This thesis will base the research methodology on Denning [3] which consists of three paradigms. Only the two first paradigms are relevant for this thesis: (1) *Engineering method of design* and (2) *experimental scientific method of abstraction of a phenomenon*. These two paradigms are combined and embedded in the research methodology described by Solheim and Stølen [2] in order to assemble a research methodology that is useful for the research to be conducted:

1. **Problem Analysis** – identify needs for a new or improved artefact. The following steps based on Denning's framework will be included here:

- a. Form a hypothesis
- b. State requirements.
- c. Specify an approach and make predictions. Since requirements stated are conceptual requirements for the solution, these will also serve as basis for predictions or success criteria here.
- d. Specification of contribution.

A literature review is conducted in order to identify needs. In order to make probable that the needs identified don't have an existing solution, state-of-the-art technologies are analysed. An evaluation of these solutions will be given with regards to the requirements specified.

2. **Innovation** – an attempt to develop artefacts that fulfil the needs identified in the problem analysis is made. The following steps based on Denning's framework will be included here:

- a. Design and implement a solution. In this thesis a solution will consist of both a metamodel and a tool, jointly called PROSERVE.

Based upon the requirements and specifications from the problem analysis the following artefacts will be realized:

- a. A metamodel including a scheme for clarifying context of services in a single model and a solution for automated identification from business process models.
- b. A service modelling tool supporting clarification of service context and automated service identification.
- c. A mechanism for scaling the visual representation of the internal structure of services in service modelling.

Artefact b) is dependent on artefact a), i.e. the tool will prove that the requirements of the theoretical solution are fulfilled. The tool will also provide a proof-of-concept for the artefact c).

3. **Validation of results:** If positive results from validation of predictions and requirements are obtained, it can be argued that the artefact satisfies the needs. The following steps based on Denning's framework will be included here:

- a. Design of an experiment, test of the solution and collection of results.
- b. Analysis and interpretation of results.

The purpose of validating the results is to confirm that the solutions actually fulfil the requirements and predication made. A positive outcome of the validation of the results strengthens the hypotheses, but it does not prove that it is correct. On the other hand, a negative outcome weakens the hypotheses and new iterations of the research process may be required. A case study is outlined to create an experiment on which the solution will be tested. This case study is an extension of the introductory case study in the problem analysis.

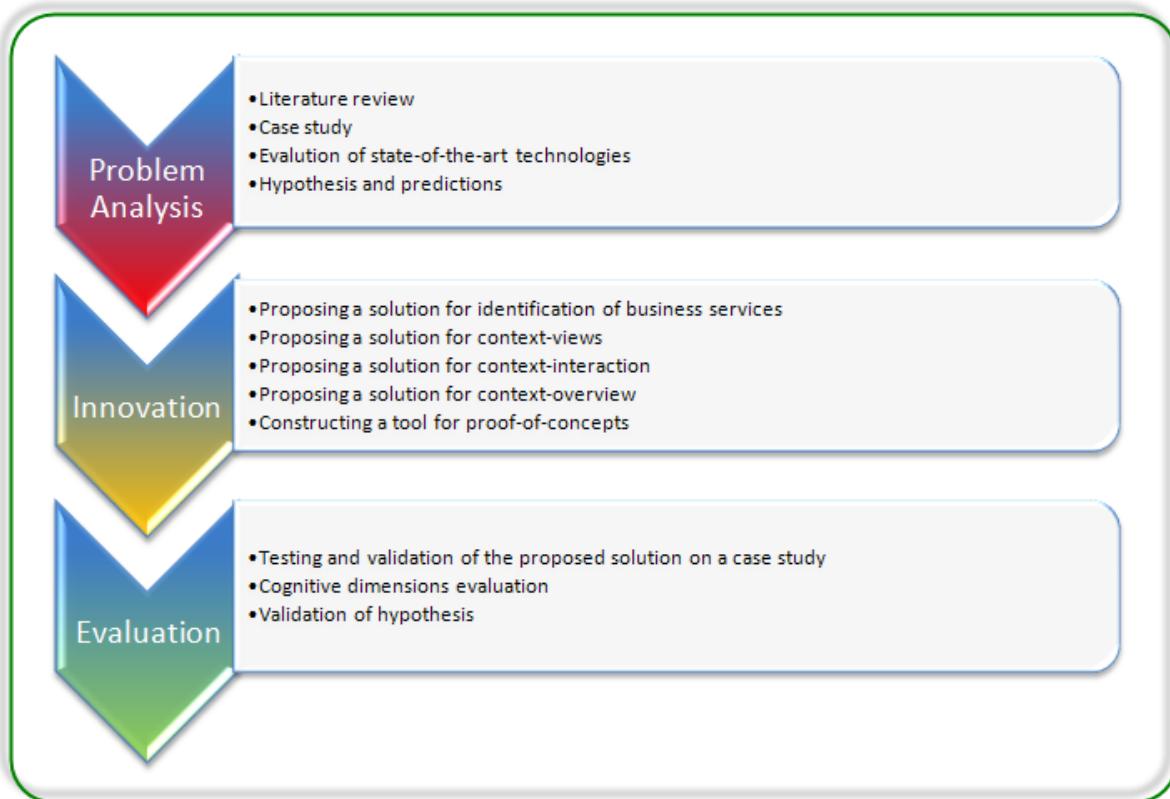


Figure 2: Use of Research Method in the Thesis Work.

For the sections on case studies and future research the author's experience from work at Ullevål University Hospital is used as input. In addition, informal interviews in the organization are conducted to collect information.

1.4. Outline of the Thesis

Figure 3 outlines the structure of this thesis. An overview on the content for the remainder for the thesis is given below.

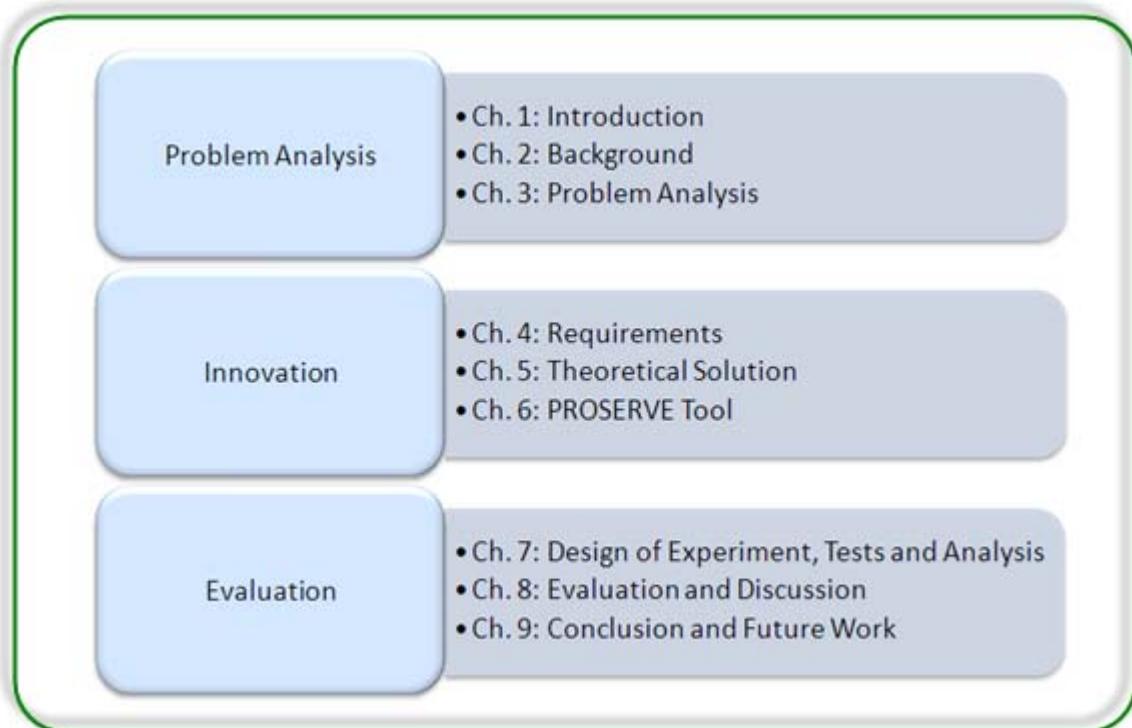


Figure 3: Structure of Thesis.

Chapter 2 (Background) outlines the background and context for the research conducted in this thesis.

Chapter 3 (Problem Analysis) consists of a literature review that aims at uncovering research gaps and needs for developing new artefacts. Moreover, a case study from health care is conducted in order to identify needs and exemplify the proposed solution. A hypothesis is stated and an approach for finding a solution is outlined. Finally, predictions are stated as success criteria for the proposed solution.

Chapter 4 (Requirements) provides sets of requirements for a service context scheme and the PROSERVE tool. These are derived from the needs identified in the problem analysis. Finally, an analysis of state-of-the-art solutions is carried out to support existence of the identified needs.

Chapter 5 (Theoretical Solution) synthesizes PROSERVE metamodel which includes a scheme for service context and a solution for identifying services from business process models. PROSERVE is tailored for the needs identified and is not a full-blown service modelling solution.

Chapter 6 (The PROSERVE Tool) presents notation, architectural design and realization of a service modelling tool with context support and automated identification of services from business process models. This tool will provide a proof-of-concept for parts of the theoretical solution.

Chapter 7 (Design of Experiment, Tests and Analysis) outlines an extension of the case study outlined in the problem analysis. The proposed metamodel and tool is tested on this case study. A cognitive dimensions analysis is also conducted to evaluate proposed contributions in the notation.

Chapter 8 (Evaluation and Discussion) evaluates the solution based on the results of tests and an analysis. It is checked whether it fulfils its requirements and related predictions. The solution is then compared with the evaluation of state-of-the-art solutions in chapter 4. A discussion is carried out to align the work conducted in this thesis with the related fields of research. Finally, the hypothesis is evaluated.

Chapter 9 (Conclusion and Future Work) concludes this thesis by summarizing contributions of the work undertaken and outlines suggestions for future work.

Chapter 2. Background

The purpose of this chapter is to outline background for the work to be carried out in this thesis. Key concepts and frameworks that are used for finding a solution are introduced here. First, service science is presented in section 2.1 to give a holistic overview over the notion of services in different domains. In section 2.2 Model-driven Architecture is explained as a means for separating concerns. Section 2.3 describes the notion of enterprise architecture, while section 2.4 depicts one of the areas in enterprise architecture: Business process modelling. Finally, Service-oriented Architecture is explained in section 2.5 with examples of frameworks and methodologies.

2.1. Service Science

Service Science is an evolving discipline that aims to provide theory and practice around service innovation [4]. It is motivated from the growth of the service sector which today accounts for the most of the world's economic activity. According to Spohrer et al, the basic category to be considered is a service system in which entities exchange performance of beneficial action composed of people and technologies [4].

Being a new area of research, it lacks general theory of services with well defined questions, tools, methods and practical implications for society. An interdisciplinary approach from the industry to provide solutions to these needs is called Service Science, Management and Engineering (SSME) which aims at bringing together ongoing work in computer science, operations research, industrial engineering, business strategy, management sciences, social and cognitive sciences, and legal sciences to develop the skills required in a services-led economy [5]. As a result, the approach needs to integrate business, information technology and humans [4, 5].

An important contribution to the Service Science is made from the marketing side by Vargo and Lusch in [6]. They define service as:

“The application of specialized competences (knowledge and skills) through deeds, processes and performances for the benefit of another entity or the entity itself” [6].

Similar definitions describe services in other domains. The main point in these definitions is value creation between entities. The notion of service in marketing is abstract.

Vargo and Lusch use the same abstract definition for services provided by IT. In cases where competences can be reduced to a list of instructions that can be communicated, services can be outsourced to IT to create self-services, for example “tell me”, “show me”, “do it for me” types of services [6].

2.2. Model-Driven Architecture

MDA is believed to boost as the prominent methodology in service-led economies as a result of the increased interests of models and componentization of IT and business [7].

Model-Driven Architecture (MDA) is an OMG¹ adapted framework for approaching software development by the use of models to direct the course of understanding, design, construction,

¹ Object Management Group: <http://www.omg.org>

deployment, operation, maintenance and modification [8]. The core element in MDA is the model which is defined as a description of a system and its environment for some purpose.

One of its main goals is to achieve independence from particular vendors and specific technology platforms. This is approached by describing a system and its environment through models at different levels of abstraction. At the top level, a Computation Independent Model (CIM) focuses on the environment and the requirements for a system. In the level below, a Platform Independent Model (PIM) describes the part of the system that does not change from one platform to another. Finally, a Platform Specific Model (PSM) is a specification of a systems usage of a specific technology.

These models exist independently, but they are related to each other through model transformation, which is the process of converting one model to another model. In MDA, a PSM may be created via a model transformation from a PIM. There is also an increasing focus on transforming CIM models to PIM or PSM models in order to formally couple business requirements and software development.

Throughout this thesis the MDA concept of views will be used to separate concerns. A view is an abstraction that enables a particular focus on particular concerns of a system.

2.2.1. Metamodels

In order to create a modelling language, semantics have to be defined [9, 10], i.e. the meaning of its elements.

Metamodelling is a core concept for defining model languages [9]. A metamodel is a special kind of model that specifies the abstract syntax of a modelling language [8], i.e. the grammar of the language. It describes in an abstract form the kinds of elements that make up the language, and the rules for how these elements may be combined [11]. MDA uses Meta-Object Facility (MOF) [12] as the means for meta-modelling.

The MOF metadata architecture consists of four layers where a layer with lower abstraction conforms to the abstraction layer above it, i.e. a meta-level provides the abstract syntax for the model at the lower level (**Figure 4**). The MOF architecture can be limited to four levels since the meta-metamodel at the top is reflexive.

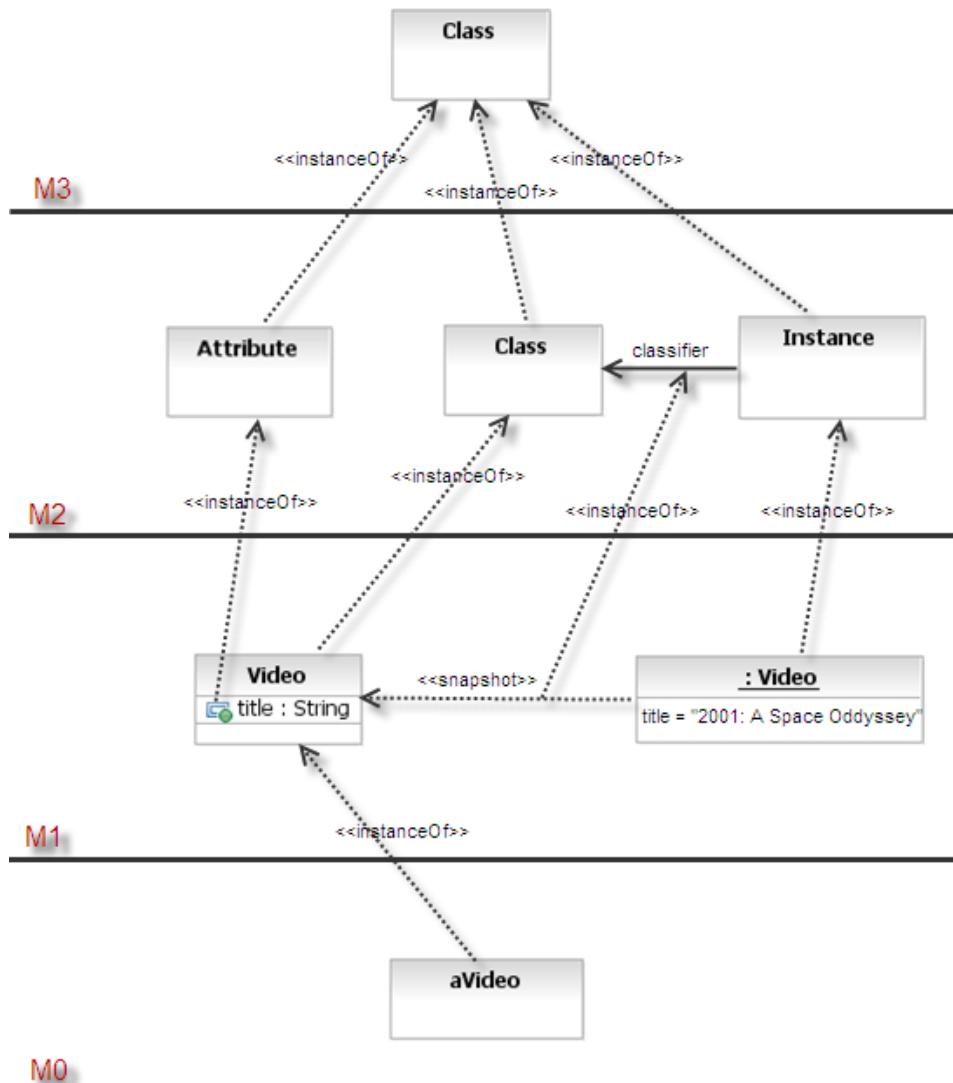


Figure 4: Concept of Layered Metamodel Hierarchy. Adapted from [13].

2.2.2. Models and Diagrams

The syntax of a modelling language consists of abstract and concrete syntax [9]. For graphical modelling languages, the concrete syntax is the graphical notation, i.e. graphical representation² of elements in the model. A concrete syntax can be presented in a diagram. Visual elements in the diagram are mapped to instances in the underlying model. A diagram is then by definition not a model.

In this thesis, the concrete syntax is referred to as notation. A view on the other hand may embrace multiple diagrams that support a specific viewpoint.

2.2.3. Model Transformation

Model transformation can be defined as creating an instance of a relationship between the input and the output domains [11]. In MDA, transformation rules can be created through mappings between entities in the source metamodel and the target metamodel. An alternative is to create model-to-text transformation where a source metamodel is used to create

² A concrete syntax may also be a textual.

transformation rules which, enables a model to be transformed into a linearized text representation [14].

Model transformation can be divided into three categories:

- A vertical transformation is defined as an instance of a relationship between input specifications and output specifications at different levels of abstractions. *Refinement* is a transformation that lowers the level of abstraction, while *abstraction* is a transformation that raises the level of abstraction.
- A horizontal transformation is defined as an instance of a relationship between input specifications and output specifications at the same level of abstraction, for example by refactoring or delocalization. The input and the output domains may be the same.
- An oblique transformation is a transformation that combines horizontal and vertical actions.

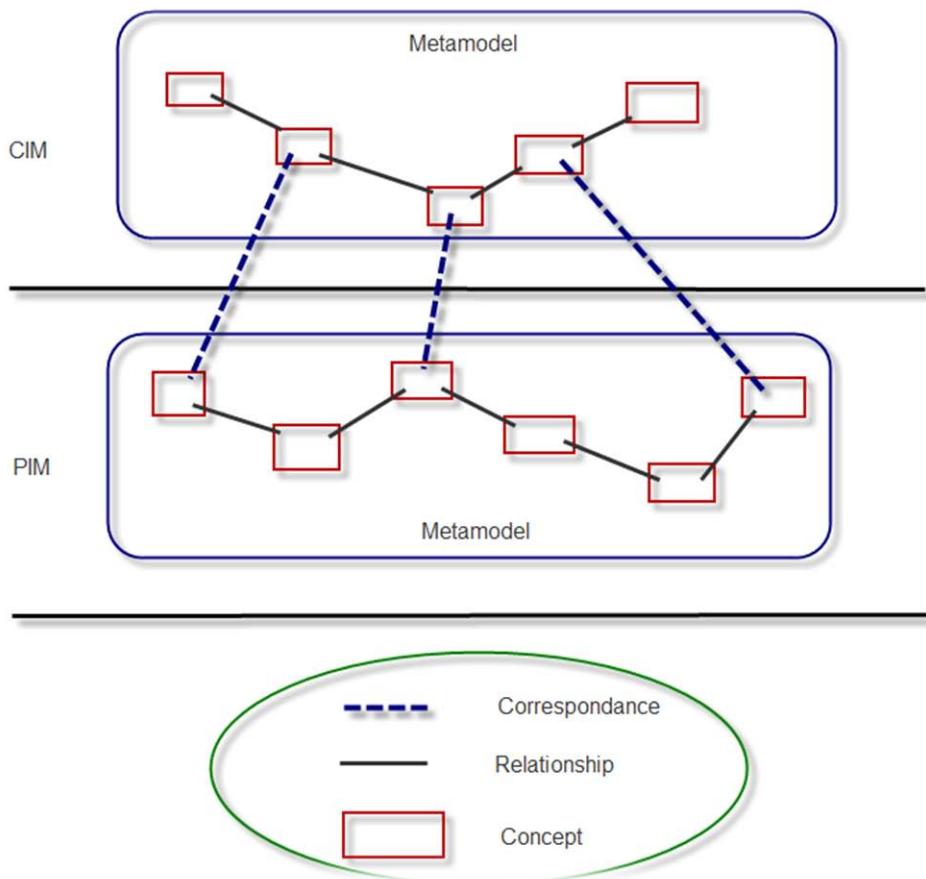


Figure 5: Mapping of Concepts between Different Levels of Abstraction. Adapted from [15].

2.3. Enterprise Architecture

Enterprise Architecture (EA) is a model-based [16] and integrated approach to business and IT that describes architecture at the level of an entire organization [17]. The notion of EA can be explained by decomposing it into its individual terms. According to the IEEE Standard 1471-2000³, architecture is “*the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment and the principle guiding its design and evolution*” [17]. The system in this context is the enterprise which is

³ <http://www.iso-architecture.org/ieee-1471/>

defined by Open Group⁴ as “any collection of organizations that has a common set of goals / or a single bottom line” [17].

Ullberg’s description of EA in [16] is limited to *models* in IT management and IT decision making. On the other hand, Lankhorst describes EA more generically as methods and principles used in the design and realization of an enterprise’s organizational structure and business processes. A key challenge in EA is to bring together information from different domains or dimensions in an enterprise to improve communication and decision making.

EA undergoes a life-cycle where ideas are realized and changes to the system and the architecture are made iteratively.

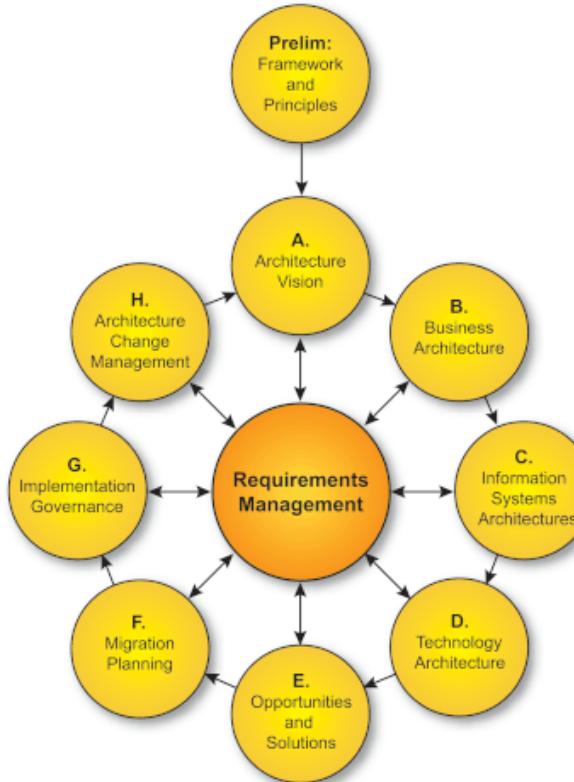


Figure 6: TOGAF Architecture Development Cycle [18].

There are several methodologies and frameworks for EA. In Zachman, which is a widely used EA framework, dimensions of the enterprise are based on roles intersecting with product abstractions in a matrix.

The Open Group Architecture Framework (TOGAF) is both a framework and method for developing an EA. Architecture Development Method (ADM) in TOGAF is an iterative process of phases in which the EA is created. The main architecture dimensions are business architecture, information systems architecture and technology architecture.

For this thesis, business architecture of EA is of particular interest. Business modelling captures strategic, tactical and operational abstractions that serve as a means to understand and improve a business [19]. Models of the business are used to represent the *business architecture* of an enterprise. The architecture provides an overview of significant parts of the

⁴ <http://www.opengroup.org/>

business in terms of its products, services, processes, organizational structure and locations. Models can be utilized to provide various views or perspectives upon the business, e.g. business process view or organizational view. In addition, they can be aligned to IT and lead to focused deliveries from IT, and serve as requirement specification from the business side.

2.4. Business Process Modelling

The *verb* process that means to handle [20], i.e. some kind of action or activity. A business process is a set of activities that have to be performed in order to achieve a specific result. Business Process Modelling (BPM) refers to the design of business processes. In BPM, activities are modelled sequentially, i.e. a specific ordering of them is required. Business Process Management (also abbreviated BPM)⁵ describes the execution of business processes as well.

The major BPM standards are based on Pi-calculus and Petri network (net) [20]. Pi-calculus is an advanced algebraic system for defining concurrent and communicating processes. Since it requires a high level of mathematical training to understand, it is not suitable for most business analysts and software developers. Petri net, on the other hand, is a formal graphical language. It uses higher abstractions than Pi-calculus such as place, transition, token and arc. A key concept in Petri net is the notion of token passing to describe the semantics of control flow.

BPM has been used to depict a sequence of work from person to person within an organization. Today, it is also used to automate processes. Workflow is defined as “*the automation of business processes, in whole or part, during which documents, information or tasks are passed from one participant to another one for action, according to a set of procedural rules*” [21]. In order to achieve this, executable business modelling languages, e.g. BPEL and XLANG, are used to orchestrate activities.

Workflow Management Systems (WFMS) is made up of software components that execute business process definitions as workflow instances and control their interaction with participants and applications. Manual activities, i.e. activities that can't be automated are not part of WFMS, but they can still be part of the process definition in a business process model. This can be the case for BPMN process definitions. In WFMS, workflow participants, typically humans, have a list of work items which represents the work that should be processed. The work items are passed on from one participant to another for actions until the workflow is completed. From a service view, WFMS can in this way execute the internal process of a service. Interactions between the WFMS and participants (employees) may also be regarded as services.

BPM is related to the notion of service-oriented architecture in the sense that software services may be identified from business processes, e.g. as in IBM's SOMA methodology [19]. It may also be an artefact for orchestration of software services in business process management.

For this thesis, BPM is only considered in the context of enterprise business modelling, i.e. models that belong at the CIM level in the MDA stack.

Business Process Definition Metamodel is ongoing standardization work from OMG that attempts to define a common metamodel for business process modelling independent of

⁵ BPM means Business Process Modelling in this thesis unless otherwise stated.

notation and methodology in a MOF metamodel [22]. This is an effort to abstract commonalities between similar business process languages such as BPMN and UML activity diagrams and to create support for MDA.

According to Havey, there are two good graphical modelling notations: BPMN and UML activity diagram, where BPMN pulls the longest straw since it is more expressive [20].

2.4.1. Business Process Modeling Notation (BPMN)

BPMN is a graphical flowchart language that can be used by business analysts or developers to build business process diagrams (BPDs) [20]. It was originally developed by the Business Process Management Initiative (BPMI), but it is now merged into Object Management Group (OMG) who is responsible for the standardization. The current version is BPMN 1.1.

In BPMN, flow objects (activities, events gateways and sequence flow) are the main elements that define the underlying structure and behaviour of the process. Basic elements that are believed to form meaningful relationships to a service representation are outlined in **Table 1**.

Activity	Description
Event	An event is something that affects the flow of the process and usually has a cause or an impact. They are categorized by the stage that they occur in the process and by type.
Activity	Activities are steps in a process that perform work. They come in two flavours. A <i>task</i> is an atomic activity that performs a single action, while a <i>process</i> is a compound activity that has its own set of BPMN constructs. Since processes are hierarchical, they are called sub-process.
Sequence Flow	A sequence flow shows the order that activities will be performed in a process. It connects the source and the target element.
Gateway	A gateway is an element to model split and joins patterns, i.e. it branches and merges paths in a process.
Pool	A pool can be used to represent a participant of the activities it contains. It often represents a company.
Lane	A lane represents a sub-partition of the participant. It often represents a unit within the company.

Table 1: BPMN Entities.

BPMN distinguishes between three different types of process:

- Private Processes: These processes are internal to an organization. They are sometimes referred to as *orchestrations or workflows*.
- Abstract Processes: These are interactions between a private process and another process or participant.
- Collaboration Processes: These are interactions between two or more business entities. Only participants and their interactions are of interest in these processes.

2.5. Service-Oriented Architecture (SOA)

SOA is a software development paradigm that has been around for a few years. SOA has many definitions. Some focus on particular software technologies, e.g. web services, while others try to define it more conceptually, e.g. OASIS SOA reference model (OSOAR).

It is an important means to reduce complexity in software systems [23]. In this perspective SOA is used as a software architectural pattern, e.g. for separation of concerns. SOA is now coupled to business and includes methodologies for deriving software components that fulfil business functions or services, e.g. SOMA. It has blurry boundaries to business architecture and may even be seamlessly integrated with it in vendor specific solutions. Although concepts in SOA are used for a solution in the business domain in this thesis, SOA itself is regarded as a paradigm for software architectures.

In the core of SOA is the service metaphor. Definitions of the term *service* are not unified and usually depend on the context and possibly on the technology considered. The use of the service metaphor has made SOA interesting for the service science initiative where it is considered to go along with the notion of service systems [24].

OSOAR provides a broad and conceptual description of SOA and is outlined here with the purpose of giving a basic understanding of the concept. In order to relate SOA to services modelling, UPMS is described.

2.5.1. OASIS SOA Reference Model (OSOAR)

The goal of the reference model is to provide a common understanding of SOA concepts and their relationships. It is an abstract framework in the sense that it explains entities and relationships between them conceptually and may serve as a base for development of standards for SOA. SOA is here defined as:

*“a paradigm for organizing and utilizing distributed **capabilities** that may be under the control of different ownership domains”* [25].

The concept of ownership is particularly interesting for this thesis as a means to provide support for context.

In OSOAR, a *service* is described as a mechanism that brings together needs and *capabilities*. In order to access capabilities, *interfaces* are provided. Capabilities are provided by a *service provider*, while a *service consumer* is someone who invokes them. Providers and consumer are jointly called *service participants*. Finally, a *service contract* is used to make agreements about the use of a service between two or more participants.

2.5.2. UML Profile and Metamodel for Services (UPMS)

The purpose of UPMS is to enable design of services within a service-oriented architecture and to fit into an overall model-driven approach [26]. It embraces SOA as both a business and a technology concept where participants are networking through services [27]. In this sense it provides a broader scope than OSOAR, which only claims to consider the software architecture view [25]. However, UPMS leverages on and conforms to OSOAR concepts.

A significant difference from OSOAR is that UPMS acknowledges that a participant can have both capabilities and needs. A participant’s needs are expressed as requisitions. In addition, UPMS describes service operations as a part of a service. However, the concept of service operation is tightly coupled with UML semantics in the reviewed submission.

UPMS is an ongoing RFP⁶ in Object Management Group (OMG). This thesis considers the submission from IBM and others from December 2007 [26]. The current submission from

⁶ Request for Proposal.

August 2008 is named Service-oriented architecture Modeling Language (SoaML). A new version will be submitted in November 2008. In the next version, the current definition of service will be redefined as a service point, while requisition will be redefined as a request point⁷.

2.5.3. SOMA

Service-Oriented Modelling and Architecture (SOMA) [19] from IBM is an example of a methodology for SOA. A service is here described as a software resource that executes a repeatable task. SOMA connects the business domain and the software domain. The view on the business domain comprises functional areas of people, processes and technology. SOMA does not, however, provide a methodology for business architecture.

In SOMA, a service model is defined as "... a model of the core elements of a service oriented architecture (SOA)" [19]. On one side it is related to models that capture requirements (business models and system use case models), while on the other side it is linked to implementation models. In this way it represents a means to couple business needs to software capabilities. However, service models are abstractions of the IT services and do not encompass other kinds of services, e.g. services provided by humans. The purpose of the model is to (1) identify candidate services and make decisions about which services to expose, (2) specify contracts between providers and consumers of services, and finally (3) associate services with the components needed to realize these services.

2.6. Summary

In this chapter, service science was described as the embracing theory from where the needs for this thesis are identified. This theory was then linked to MDA. Concepts in MDA that support separation of concerns were explained. Examples were metamodelling, model transformation and abstractions-layers. BPM was then described and its relationship to SOA was explained. Finally, UPMS and SOMA were introduced as a modelling standard for services and a methodology for SOA respectively.

⁷ Source: SoaML project manager: Dr. Arne-Jørgen Berre.

Chapter 3. Problem Analysis

The purpose of the problem analysis is to identify research gaps and needs for developing new artefacts. First, a case study is carried out in section 3.1 which covers a case for sending appointment reminders in health care. A literature review in section 3.2 is conducted in order to analyse the notion of service and its relationship to business processes. Section 3.3 outlines the needs that are identified. They are used to derive a hypothesis in section 3.4, which again is used to derive predictions about the solution in section 3.5. Finally, an approach for finding a solution is described in section 3.6 based on this problem analysis and the background theory presented in Chapter 2.

3.1. A Case Study: Health Care Appointment Reminders for Patients⁸

In order to anchor the need for research, a case study is outlined here to exemplify problems in the area of study.

Ullevål University Hospital⁹ (UUS) is Norway's largest hospital [28]. It has 9500 employees and 580,000 patients each year. The total budget for 2004 was approximately 5.2 billion NOK and is funded by the government. A regional health enterprise has statutory responsibility for ensuring provision of health services offered by UUS.

An important goal for the hospital is to reduce its expenses. Optimization of resources is one approach for achieving this. Each time an appointment is scheduled at the hospital, resources are allocated. When a patient misses an appointment, these resources are not utilized. In order to optimize use of resources and throughput of patients, UUS decided to remind patients about their appointments.

An appointment reminder service for patients (ARSP) is conceptually not bound to any specific technology. Although it represents a service for patients, its primary target is clinical departments that want to remind their patients about the appointments. The following sequence captures activities that surround the planned service:

1. A general practitioner refers a patient to the hospital.
2. A specialist at the hospital assesses the referral.
3. A clinical department at the hospital grants the patient an appointment
4. A clinical department sends an appointment letter to the patient
5. ***The hospital reminds the patient about the appointment (ARSP)***
6. The appointment is processed.

It is predicted that the patient will contact the hospital for different reasons. These events are called exceptions and include:

- Patient feels that the time is incorrect.
- Patient has never been informed about this appointment.
- Patient is unable to attend.
- Patient does not know where the appointment is.

⁸ Based on experiences with implementation of SMS reminders for patients at Ullevål University Hospital

⁹ <http://www.ullevaal.no>

The process that fulfils the ARSP may be modelled in a business process diagram using BPMN notation.

A central unit, the logistics department, at the hospital is given the responsibility of providing this service. It is up to each clinical department whether they want to subscribe to the service or not and individual service contracts have to be made. Contracts need to specify what is delivered and who is responsible for providing the service. In some cases the contract may also have to specify how the service is delivered, e.g. that a set of rules are processed before a patient is reminded.

Ideally, the name of a service should describe what it delivers. However, service delivery often needs to support different needs that typically belong within the service umbrella, e.g. the radiology department may want to add additional information about pre-medication before an examination while the oncology department may want to remind their patients two times. In order to describe both these deliveries within the same service, another artefact is needed.

Before technology decisions are made there is a need to represent the service conceptually in order to focus on the business value and understanding of it rather than the actual realization of it. Such a representation can give the hospital better grounds for making decisions about how it should be realized.

A decision is made to use Short Message Service (SMS) to issue reminders to patients based on booking information in the booking system: Two days before an appointment, a reminder is issued automatically from the hospitals IT infrastructure. At this stage, there is a need to tag the conceptual ARSP as a service that will be realized by IT.

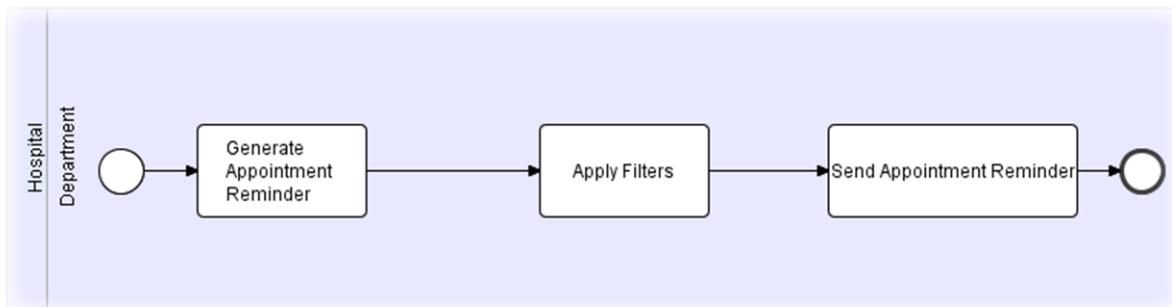


Figure 7: Send Sub-Process.

So far the new service does not seem to require organizational or human labour effort. However, privacy concerns regulated by law constrain the content of the message that is sent out. The hospital cannot inform the patient at which department the appointment is. As a consequence, all responses *from* the patient must be routed through the central switchboard. This represents challenges in cases where the above described exceptions occur. In addition, it demands increased resources at the switchboard. So, a decision is made to create a separate service to deal with exceptions. The logistics department need to make a contract with the switchboard to be able to offer these capabilities to the departments that subscribe to the ARSP. Departments may in theory choose whether they want to include the exception service in ARSP. However, for practical reasons, the exception service has to be included for all consumers.

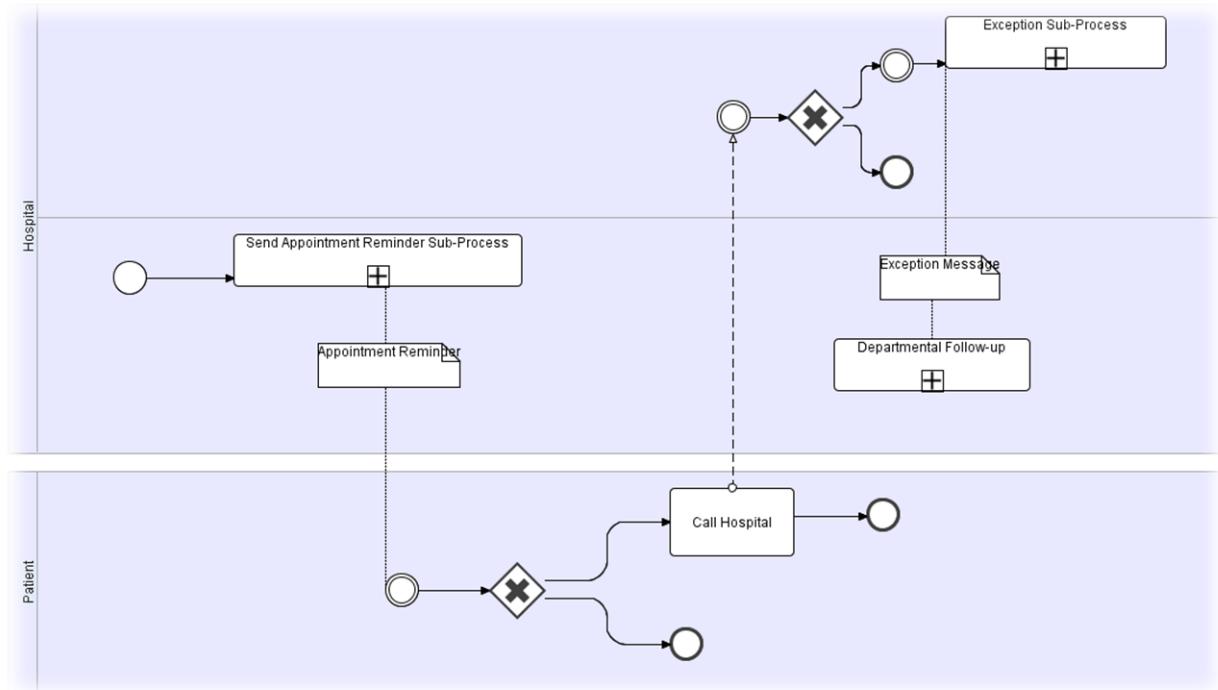


Figure 8: ARSP Main Process.

This example illustrates that the effort of saving resources in one area of an enterprise may increase expenses in other units (**Figure 9**) and even the units where the savings effort was initiated (**Figure 10**).

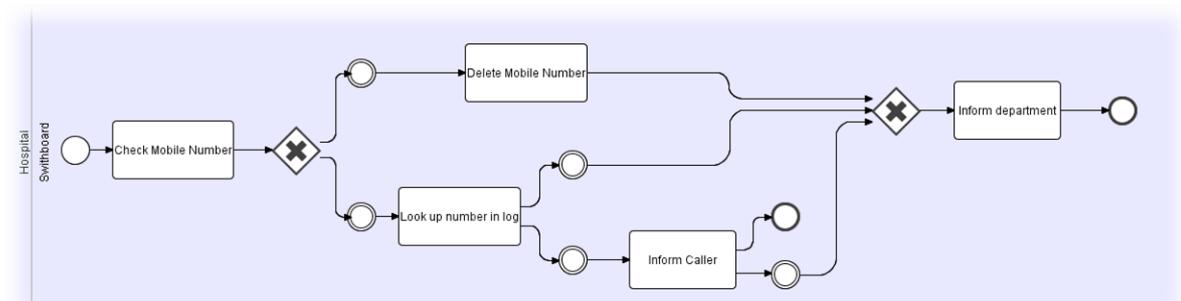


Figure 9: Exception Sub-Process

The Exception Sub-Process triggers the need for departmental follow-up, so that that patient receives the information that is required.

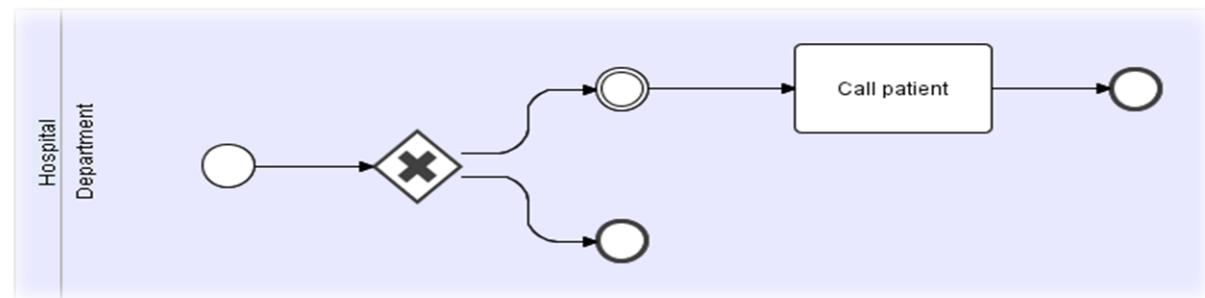


Figure 10: Departmental-Follow-Up Sub-Process

It may therefore be useful to describe supporting activities from various units as services and make agreements between them on responsibilities, quality and funding.

From a strategical point of view, the hospital needs an overview of which services are delivered by *what*. As an example, an alternative approach to sending SMS could be to call patients. This would then be a service delivered by humans. It is assumed that human services are more costly than IT services, so an overview over whether services are delivered by IT or humans would be useful to discover services that could be outsourced to IT. An overview would also facilitate identification of redundant services in the organization.

Strategically, it is also important to know which services are offered to the outside world and dependencies to external entities. In this example, the ARSP also represents a service that is provided across the organizational border, i.e. to the patient. So conceptually it serves two needs: A clinical department's need to reduce the missed appointment and a patient's need to be reminded. In this setting, it is the department that offers the service to the patient. To represent that the clinical department is responsible for delivering the service to the patient, a new service must be created to respond to the need from the patient (**Figure 11**).

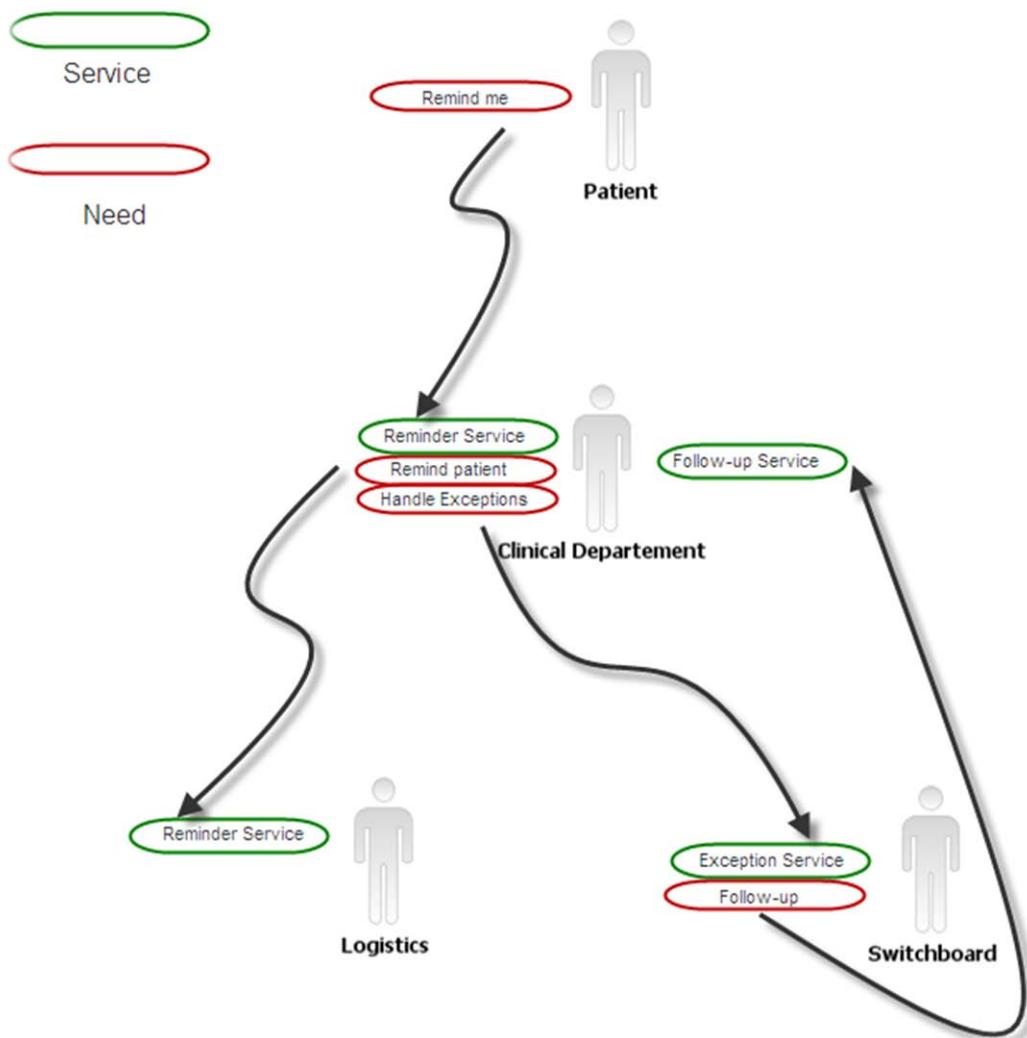


Figure 11: Illustration of Service Delivery.

The IT department at the hospital delivers software capabilities to the enterprise as software services. In theory, there is an opportunity to trace conceptual services from a business point of view to the actual specifications of the software services. However, there is no solution for how to do this tracing.

Another challenge for the hospital is to rapidly identify services from business processes. The core business is expressed in business process models. In order to support service-orientation of the business, these processes need to be translated into service definitions. Identifying these services at the business level may also contribute to better correspondence between these services and SOA services.

Finally, when developing services it is necessary to acquire an overview of other existing services in order to avoid redundancy and to promote re-use. This requires not only a helicopter view of services, but also a mechanism to zoom in on details of a service to see the content of services delivery and service fulfilment.

3.2. Literature Review

An initial assumption for the literature review is that the service term needs clarification with regards to its context, its internal structure and how they interact.

3.2.1. Alignment of SOA Services, Customer-oriented Services and E-services.

Service has become an important concept for business and software development. In the world economies there is a shift from goods-centric view towards a service-oriented view [6]. The use of the service metaphor from different perspectives has motivated the emergence of services science as an academic initiative from the industry. This is an effort to create a multidisciplinary field that changes the focus from a technology centric view to a holistic view that encompasses both technology and business [1].

The meaning of the term service is extremely broad [29]. In this section, an effort is made to align the meanings of SOA services, business- or customer-oriented services and E-services.

Kiyomizu has classified services into different intersecting categories [29]:

- Intellectual and spiritual services are abstract ideas, concepts and principles. They constitute the basis for the higher level services.
- Behavioural services are services that create an atmosphere for client interaction, e.g. gestures and expressions.
- Business and operational services provide intangible goods and direct economic value.

The latter category covers the problem area in this thesis. These services are sub-divided into more specific categories ranging from those dealing with human/social capacities, tangible goods, to information services not provided by human beings.

The customer plays an active role in the new service-oriented view in marketing. According to Vargo [6] goods are no longer regarded as the primary unit of exchange between the business and the customer. People acquire benefits of specialized competences (knowledge and skills), called operant resources. Tangible goods (operand resources) only act as transmitters of these operant resources. As a result, the customer is seen as being a co-producer of the service and thereby also the co-producer of the value of it. This implies that

the customers are no longer acted on by the business, but they are active participants in relational exchanges and co-production. From this customer-oriented perspective, in order for something to be a service it should be consumed or be consumable by human beings. Such a view would exclude activities that are not directly of value for people from being classified as services. If the value of operant resources is based on cognitive functions, an electronic consumer will not be able to compute the value of these and is thus not a co-producer of value. However, in order to extend the applicability of this theory it may be argued that an operant resource represents value for an electronic entity since it may require this to fulfil a given goal. Since Vargo's context is marketing, he does not explicitly debate electronic consumption of operant resources. Despite that, Vargo's definition of services is broad and applicable for both human and non-human activities through his definition of service as a kind of action, performance or promise that is exchanged for value [4].

Consumer behaviour in cyberspace can be called e-services [30], i.e. online capabilities required by humans accessed through technology. The term e-service has several definitions ranging from technology specific descriptions to broad meanings that encompass service product, service environment and service delivery [31]. E-service can be defined widely as deeds, efforts or performances whose delivery is mediated by information technology [32]. Froehle and Roth's *modes of face-to-screen customer contact* encompass this kind of service interaction (see **Figure 12**) [33]. They describe how technology and humans interact in customer-oriented service interaction. The two main modes of customer contact here are *face-to-face* and *face-to-screen*.

The literature suggests that e-services are services consumed by human customers, which implies that an electronic service that exists for the benefit of another electronic service is not an e-service. Van Riel distinguishes between these services by proposing five components in e-service: (1) the core service, (2) facilitating services, (3) supporting services, (4) complementary services and (5) the user interface [32]. None of these categories encompass the concept of service interaction between two electronics entities. A reason for not finding a category for this type of interaction in the e-service literature reviewed may be that they focus on the business perspective and not on how services are represented internally in technology.

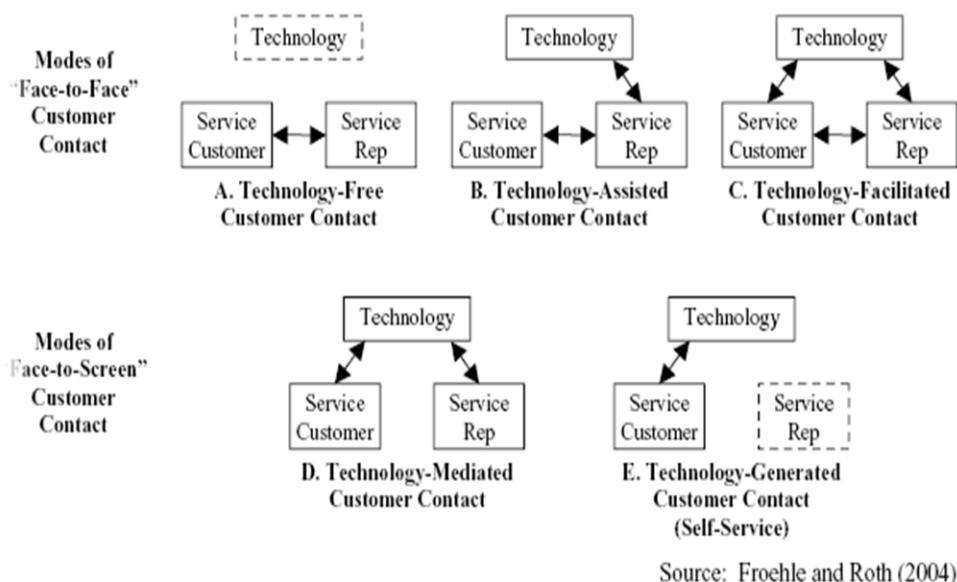


Figure 12: Service Interaction [33].

Service-Oriented Architecture is a paradigm in IT that seeks to enable business needs to be connected to software architectures [25, 34] via concepts of needs and capabilities in the service metaphor [25]. In this regard, software services may also be called business services since they provide capabilities for business purposes. However, SOA is also used as a software architectural pattern that is not necessarily linked to the concept of service in a business-oriented view. Before the emergence of SOA, Vissers and Logrippo recognized the service concept as a means to achieve separation of concern and adequate abstraction in distributed-systems design [35]. A service design pattern regards each system as a system itself in a recursive manner into different levels of abstraction. These decomposed systems can then be reassembled in different ways to create new systems. Interactions between one system and its environment occur over interfaces that do not disclose the internal organization of the system [35].

A current problem of SOA is that it has diffused into the business domain, i.e. business architecture, without reconciliation of the broader perspective on services that includes technology independent service definitions and services provided by humans. Nor does it describe how conceptual human services are aligned with conceptual software services at the level of business architecture. As a consequence, SOA diffusion in business architecture may promote technology-driven solutions for the enterprise rather than a business-needs-driven approach for solutions. According to Sinderen, it is technology that drives SOA and many authors associate SOA only with Web services, although there is a consensus that there are several alternative ways to implement it [35].

SOA's coupling to the business is emphasised by Datz who claims that SOA focuses on the business processes [36]. Although it may be claimed that applying SOA in higher levels of an enterprise is a way to service-orient the business [37], it may blur the motivation. **Since both business and software concerns can be represented in a service metaphor, there is a need to clarify the concerns and the context it is used in.** The need to represent services in a way that encompasses business concerns has motivated the development of an augmented view in service models as for example the Unified Service Model (USM) [37]. The approach taken in the development of UML Profile and Metamodel for Services (UPMS) [26] is to generalize the service concepts to broaden the applicability of the model. These two approaches focus primarily on the business view and the software view respectively. USM claims to take a holistic view on services, but fails to separate the concerns. Thus, a need to separate concerns in service modelling exists.

The use of the term business service in this thesis refers to a conceptual service that is decoupled from its realization. It will therefore not be used to describe SOA services in this thesis.

Summary of Service Concerns

In the reviewed literature above, the different semantics and contexts were analysed. **Table 2** shows important findings.

	Business	E-services	Software
Main motivation	Exchange of values	Information or value exchange	Design pattern. Separation of concern.
Service Consumption	Abstract	Abstract	Concrete
Service Delivery	Abstract	Abstract or concrete,	Concrete
Participants	Human and software components	Human and software components	Software components
Role of customer	Important	Important	Not a concern

Table 2: Concerns of Services in Business and Software Domains.

3.2.2. Granularity in Services and Business Processes

The purpose of this section is to look into the service metaphor and business process with regards to their operational artefacts: operations and activities respectively.

Zimmermann, Krogdahl & Gee propose that SOA should consist of three layers of abstraction [38]:

1. The business process as set of actions or activities with defined long-term goals.
2. A service as a logical group of low-level computing operations.
3. An operation as a single logical unit of computation.

No business semantics have been put into the definition (2) and (3). From the definitions, a service is just a container for operations while an operation is the element that is responsible for the actual work.

Service granularity refers to the scope of functionality exposed by a service [39]. According to Papazoglou and Van Den Heuvel, service operations should be relatively coarse grained and reflect the requirements of the business process [39]. They suggest that service operations in a service should implement a complete business process. However, they also say that services may be used to accomplish single business tasks. Evidently, the challenge of service granularity makes it necessary to make human assessments in order define granularity of service operations wrapped in services. The challenge is enforced by the fact that business processes may be modelled at multiple levels of abstraction. As an example, a business task may be recursively decomposed into sub-processes.

In SOA, granularity of services is important to consider in order to optimize re-use of them. Atomic services are services that are meant to be reused in different business processes [40]. These are assembled in process-oriented ways to form composite services. Composing services that are made of other services, is called recursion [37], orchestration (local view) or choreography (global view) [20].

The structures in business processes have different names and meanings. Qiu describes business processes as typically consisting of a set of services [36], i.e. composite operations. While in Business Process Modeling Notation (BPMN) behaviour is called activity which can be both atomic and compound [41]. Event Process Chain (EPC) is a widely used proprietary standard used in the industry to represent business processes. Behaviour in EPC is constructed from functions which are defined as tasks or activities [42].

From a business system perspective, services consist of two subsystems [43]. In service operations (1) inputs to the service product are processed and produced, while in service delivery (2) inputs are combined and delivered to the customer in the end. Applying these terms to the case study from section 3.1, would mean that making a booking would be a service operation for the delivery of an appointment reminder.

In UPMS, services and operations represent different granularities and semantics, e.g. operations are more fine-grained than the notion of service. Above it is suggested that a service can be a container for operations. However, in UPMS a service is specified as a connection point for other services or requisitions. This metamodel definition may work for software services, but it is too limited to represent the understanding of what a service is in business and customer-orientation.

3.2.3. Service Interaction

A business process may be realized by information technology, people, or both. SOA realizes business services that can be realized in IT [36]. However, many businesses deliver heterogeneous services that need to be aligned. For instance health-care providers may provide a diagnostic service which consists of activities like receiving results from medical tests electronically, assessing the result and informing the patient about the diagnosis face-to-face. Qui provides a model for how the different services may co-exist (**Figure 13**). However, the model does not consider how services interact within the same level in the Service-Oriented Business layer, i.e. how services interact internally in the enterprise. **Thus, there may be a need to represent how services interact at this level.**

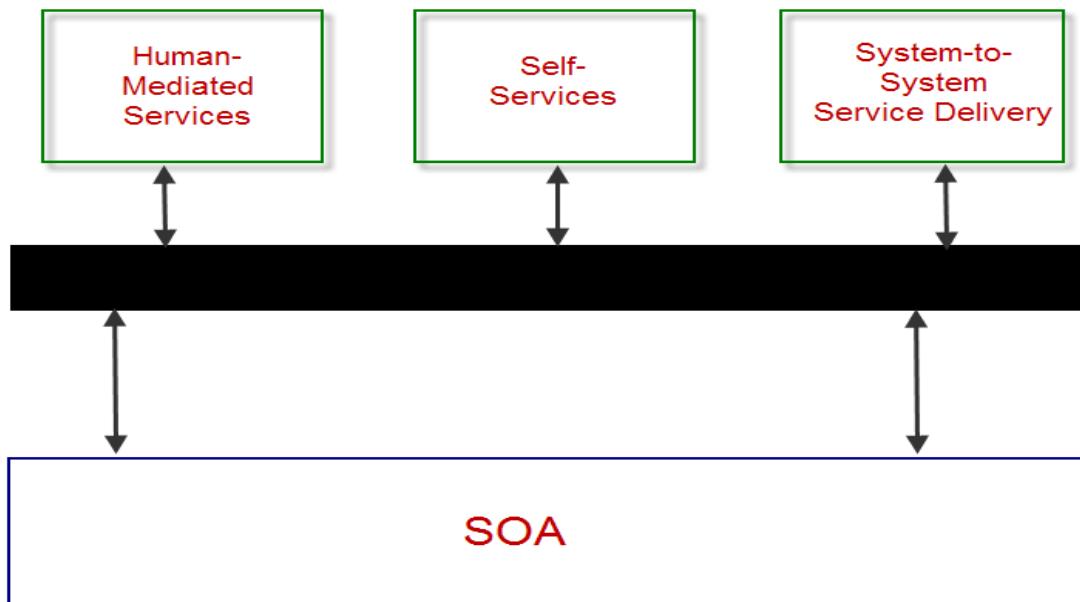


Figure 13: Service-Oriented Business. Adapted from [36]

Maglio et al describe the interaction between the service provider and the service client by specifying a service target that describes the form of intervention [44] (see **Figure 14**). Service interaction is here described from a business perspective and depicts co-production of value between service provider and service client. Service target describes the reality to be transformed by the provider in order to deliver the service, e.g. organization, products or information.

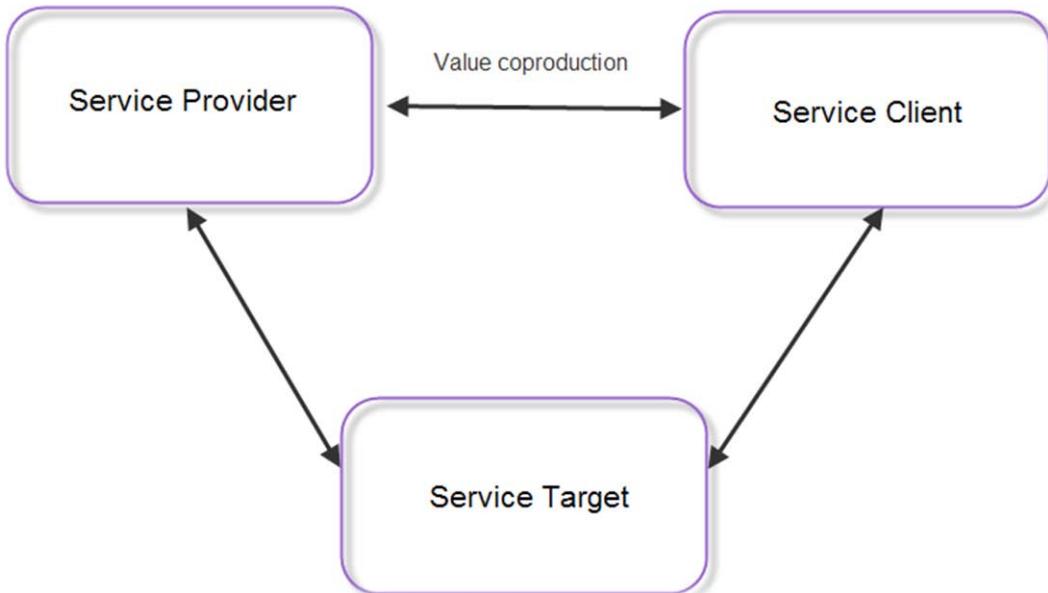


Figure 14: Service Interaction. Adapted from [44].

3.2.4. From Business Processes to Software Services

Business Process Management has emerged as a mechanism to connect business requirements to IT services and processes. It is a way of “*managing the execution of IT-supported business operations from a managerial process view rather than from a technical view*” [45]. A core challenge described by Hepp and Roman is the continuous, bi-directional translation between the business requirement view in the process segment of the business and the process execution space in the IT domain [45].

Business process models can be transformed to an executable form by manual transformation or automated transformation [36]. Manual transformation involves converting business-level documentation to IT representation. It may suffer from the cultural gap between the two domains [36]. In automated transformation, business process models can be transformed automatically into an executable representation. Business Process Execution Language for Web Services (BPEL) is such a representation. However, the business process model can only be transformed to abstract BPEL code. The generated code has to be processed by human effort to implement services and to make bindings to existing executable services. At the same time there is no formal contract between the business and the IT side and limited traceability back to the original business process model. A business service model that integrates business process models may be a step towards full integration of heterogeneous processes and representation of organizational ownerships.

For hospitals most core processes aren't possible to automate with today's technology. A key challenge within BPM is to integrate manual and executive processes [46]. Human labour that is included in the business process model may be challenging to integrate formally in an executable model. BPEL4PEOPLE is a joint effort from the industry to include representation of human activities in BPEL code [47]. The work to include human workflow in BPEL is currently an activity in OASIS [48].

3.2.5. Scalability of Services in Conceptual Modelling

A challenge in business process modelling is to provide a notation that supports scaling [49]. Both UMS and UPMS lack mechanisms in their notation in order to scale the amount of information displayed in a diagram. This may lead to severe cobweb problems or hidden-dependencies when the model is modularized, e.g. drill-downs, to limit the amount of information. It is therefore difficult for the user to get an overview of the diagram as it grows bigger. Thus, there is a need for a mechanism to represent services in a scalable way that minimizes these problems.

Semantic zooming is a principle of suppressing detailed information and displaying it in a simplified form [50] to enable better overview. This is also called abstraction. In selective abstraction a user focus on a specific region in a diagram and needs to see the details at the same time as the surroundings or context needs to be seen. A way to support this is to abstract the context while keeping the focused area unchanged. Köth and Minas explain that abstraction can be achieved by removing details from a diagram and replacing them with a visual element representing the abstraction [50]. When a need to focus in on the details of the abstraction occurs, this process must be reversed. This mechanism is an abstraction by means of diagram transformation. Abstractions are often based on hierarchical structures.

Enterprise Modelling Language (EML) uses a collapsible hierarchical tree layout to represent services and operations [49]. Here, services and operations are nested in a single hierarchy. This abstraction works for services in one configuration such as in a business process. Although the elements in the tree are called services and operations, the diagram is intended to visually represent processes. A vital point of services is re-use. This is, however, not well represented in the diagram, i.e. it is not possible to show all dependencies of a service to other services. In addition, when an abstraction is used to represent a re-usable service, the details have to be expanded misplaced from its context.

A need to display services in a way which offers an overview while at the same time enabling contextual focus on details in diagrams, has been uncovered.

3.3. Research Needs

The case study and the literature review above have identified needs for a solution with regards to conceptual service modelling. It was revealed that the hospital needs to represent a service and its relationships to other services in the organization. An overview of dependencies to both human and technology delivered services is also needed. In order for an enterprise to be adaptive, people, processes and technology should be integrated across the enterprise [36]. A service model should therefore integrate these dimensions.

Since the preferred practice to describe business at the hospital is through business process models, a solution for automatic service identification from these models is a way to rapidly populate a service model of the hospital.

Software services that are used by the hospital may be outsourced to other software services. It is therefore important to have a business description of the services that is loosely coupled to the software description of the service. At the same time it is important that dependencies can be traced both in a top-down and bottom-up way to make assessments about impacts of changing services at different abstraction levels.

Finally, the case study raised the question on how to represent services in a way that gives decision-makers an overview over the services in the organization. In the literature analysis, this was coupled to the challenge of abstraction and zooming in diagrams.

Separation of concern is a challenge associated with system modelling using multiple, overlapping viewpoints that utilize possible heterogeneous languages [51]. The literature revealed that there is a need to clarify the context in which the service metaphor is used. Different meanings of the term are summarized in **Table 2**. According to Webster's Encyclopaedia, context is "*the set of circumstances or facts that surround a particular event, situation, etc*" [52]. In this work, context is about (i) an element's surrounding artefacts (overview); (ii) its level of abstraction; (iii) how they are aligned for interaction. These types of contexts will be referred to as *context-overview*, *context-view* and *context-interaction* respectively.

Artefacts that do not belong in a defined context need to be suppressed. A given context also needs to provide a solution for how to derive artefacts into other contexts, e.g. a service in a business context may need to be derive a service that belongs in a software context. A view is a given context at a given level of abstraction. The main views needed for the hospital are a business view and a software view. In addition, an integration view may be useful for viewing artefacts from the main views in order to provide a context for reconciliation of the different concerns.

Needs Identified
Clarify the context-viewpoint for services in service modelling (context-views).
A mechanism for automatically identifying services from business processes.
A mechanism for propagating needs in the business domain to services in the software domain (context-interaction).
A mechanism to differentiate between human and software delivered services (context-interaction).
A mechanism to give an overview of services and to zoom in on details in the context they exist in (context-overview).

Table 3: Research Needs.

3.4. Hypothesis

A hypotheses specifies assumptions about a solution to a problem [2]. It cannot be proven, but can be strengthened or rejected through experimentation and testing. In research it serves as a basis for validation of results obtained. The hypothesis for this thesis is:

H1: *A business service modelling tool based on a service context scheme and a process-driven service identification will provide a means for presenting services in the context they belong to, so that SOA can be better coupled to business models through the use of the service metaphor in both domains.*

From the defintion of contexts above, context here means both context-views and context-overview. The assumption is therefore that contexts can be provided between abstraction views and within a specific view to present services with other services in a useful way.

3.5. Success Criteria

In order to validate the hypothesis, success criteria have to be made. The success criteria are predictions about the solution with regards to the hypothesis (**H1**). They describe improvements and desired effects that the solution will have on business service modelling.

Success Criterion 1 (P-1): *The tool can provide a service representation of a business process model in a way that retains the order of activities.*

The tool needs to be able to create elements that make up the service contract in the service model that includes correspondence to the order of activities in the business process model which the service is identified from.

Success Criterion 2 (P-2): *The tool can visually differentiate between services delivered by humans and services delivered by software in a business view.*

In order for business people to represent decisions about outsourcing services to IT, these have to be tagged at a business level representation. Being able to see how related services are delivered and interact, may contribute to a better view of the context of a service to be outsourced.

Success Criterion 3 (P-3): *The tool can visually provide a mechanism for delegating services from a business view to a software view.*

Service delegation is a way of translating a service in a business view to a service that belongs to a software view. It will also serve as an artefact for traceability and as a means for business people to decompose services into services with more manageable granularity.

Success Criterion 4 (P-4): *The tool can provide a software view where only services that represent software services are visualized.*

A software view will visualize service definitions that are coupled to definitions in the business view. These will be definitions that they can “hand over” to IT-people.

Success Criterion 5 (P-5): *Services in the tool can be traced bi-directionally across the domains to show dependencies.*

Bi-directional traceability enables visualization of services affected by a given change.

Success Criterion 6 (P-6): *The tool can be used to visually present a service contract between a consumer and a provider of a service in a business view.*

Visual service contracts will enable business people to view and assign responsibilities.

Success Criterion 7 (P-7): *Service model instances can be presented as selective abstractions through a service tree in diagrams.*

Selective abstraction will enable business people to view details of a service at the same time as the surroundings can be viewed, e.g. which other services it interacts with.

3.6. Approach

The solution proposed in this thesis will be referred to as PROSERVE. It aims at resolving the problems identified in section 3.3.

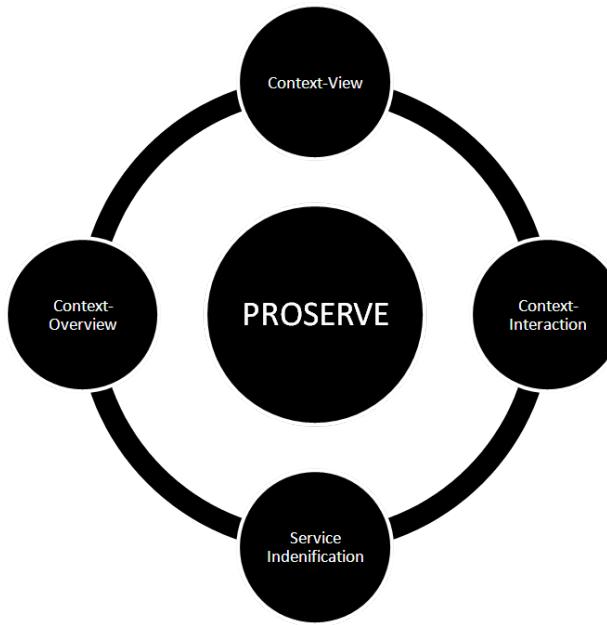


Figure 15: PROSERVE Concepts.

A model-driven (MDA) approach is chosen for creating contexts within a model and to derive artefacts between heterogeneous models. In order to support different contexts for services, views upon a shared service modelling language are attempted to be created. The Views will support for the viewpoints it is intended to support. In MDA, a viewpoint is a technique for focusing on specific concerns within a system by suppressing selected details [8].

By synthesizing from the literature review, a scheme for service context will be made based on a taxonomy for services. The scheme will provide rules for context-views and context-interaction.

A metamodel for services is constructed based on concepts from UPMS and OSOAR. The scheme is then embedded in a model. To limit the scope of the model, the focus is set on the business of services rather than service design for software development.

In order to create a solution for automatic process-driven business service identification, business process model entities are mapped to entities in the new service metamodel.

The approach for context-overview is to create collapsible hierarchical representations that can hide selected details and be used to locate related elements. A solution for context-overview is described in the notion for PROSERVE.

Finally, a tool is created to provide proof-of-concepts. **Figure 16** illustrates the context of the research work.

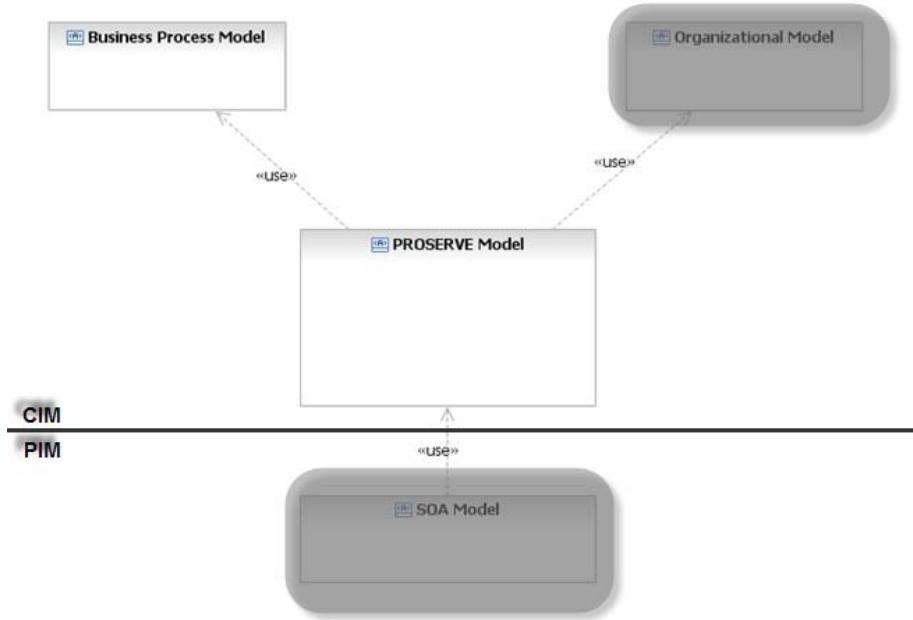


Figure 16: *Context of Work.*

3.7. Summary

In this chapter, needs for research were identified through a case study and a literature review. Within the scope of the thesis, an approach for a solution was proposed. A hypothesis was formulated in section 3.4 which will be evaluated in section 8.5, based on the derived predictions in section 3.5.

Chapter 4. Requirements

In this chapter, requirements for a metamodel and tool are derived from the needs and approach described in Chapter 3. These will serve as evaluation criteria for the solution in Chapter 8. In section 4.1, requirements for PROSERVE metamodel are described as requirements for a service context scheme. Section 4.2 defines requirements for the PROSERVE tool. Some of these requirements require a theoretical solution provided in PROSERVE metamodel. Finally, an evaluation of state-of-the-art tool frameworks and methodologies for service modelling and service engineering is carried out with regards to the tool requirements described in section 4.3. This evaluation is later used to compare the proposed solution in 8.4.

4.1. Service Context Scheme Requirements

This section outlines core requirements for the PROSERVE metamodel.

Scheme Requirement 1 (SR-1)

The scheme shall provide a solution for segmenting services and requisitions into a business service view and a software service view.

In order to separate concerns, a solution is needed to filter services into the different categories. Since services and requisitions in all views are instances of the same metamodel entity, a mechanism is needed to assign them to a specific domain in a meaningful way. This will be a solution for context-viewpoint.

Scheme Requirement 2 (SR-2)

The scheme shall provide a solution for differentiating between the natures of service delivery in the business view, i.e. whether they are provided by humans or by information-and telecommunication technology (ICT).

A mechanism is needed to ensure that services in the business domain are coupled with services in the software domain. Only services with ICT properties in the business domain can be used to derive services in the software domain.

Scheme Requirement 3 (SR-3)

The schema shall provide a solution for how services are allowed to interact (context-interaction).

A given service is not allowed to provide service delivery both to human entity and a software entity. Moreover, a human service in a business view cannot communicate directly with a software service in a software view. This is both a rule of context-viewpoint and context-interaction. It needs to communicate via a business representation of the software service meant to be consumed by humans. A proof-of-concept for context-interaction in a prototype will not be provided due to time limitations.

4.2. Tool Requirements

In this section, requirements for the PROSERVE tool will be outlined. Some of the requirements are generic tool requirements, while others are indirect requirements for the

theoretical solution as well, for example a solution for automatic process-driven business service identification.

Tool Requirement 1 (TR-1)

The tool shall be implemented in open source technology.

Realizing the tool using open source technologies will facilitate development without having to pay for development tools.

Tool Requirement 2 (TR-2)

The tool shall be metamodel-based.

A metamodel will define the abstract syntax of the service modelling language. It can be used to derive model transformation rules.

Tool Requirement 3 (TR-3)

The tool shall differentiate between abstract and concrete syntax.

Pollution of the metamodel of notational concerns creates dependencies between the underlying model and notational syntax, and should be avoided since other notations might be used in the future.

Tool Requirement 4 (TR-4)

The tool shall support service behaviour.

The tool shall be able to represent ordering of activities to support descriptions of process realization of services.

Tool Requirement 5 (TR-5)

The tool shall support automated service identification from a business process model.

The tool shall be able to populate a service model from a business process model.

Tool Requirement 6 (TR-6)

The tool shall support selective abstractions for service contracts.

By providing a means for selective abstractions, the tool will support context-overview. The user will then be able to expand details in a diagram while other details are hidden in abstractions.

Tool Requirement 7 (TR-7)

The tool shall support traceability of services.

The tool shall be able to present services that have relationships to other services. Traceability enables a user to analyse the impact of change by visualizing service dependencies.

Tool Requirement 8 (TR-8)

The tool shall support separation of context-viewpoints based on a common model.

The tool needs to provide a mechanism for displaying services from both a business context and a software context by defining different views. A business view is meant to facilitate

decisions on how services are realized, i.e. by manual work or by software, while a software view is an overview of *outsourced* service definitions that potentially can be propagated to software design models.

Tool Requirement 9 (TR-9)

The tool shall use metamodel-based model-transformation

In order to automate service identification, mapping rules should be created from the source metamodel. The tool can either use model-to-model transformation or model-to-text transformation.

4.3. Evaluation of State-of-the-art Tool Frameworks and Methodologies for Services Modelling

Tool frameworks and methodologies for service modelling are evaluated here in order to align and strengthen the needs identified in the problem analysis. The frameworks that are selected for evaluation represent state-of-the technologies within the field. Two academic and one industry framework are considered.

Component and Model –based Development Methodology¹⁰ (COMET)

COMET is a methodology from SINTEF¹¹ for developing and maintaining software products [53]. It embraces modelling from the business perspective to the software perspective via a service model. The methodology has adapted OMG standards such as BPMN, BMM¹² and UPMS. Tool support is implemented in IBM's Rational Software Modeller.

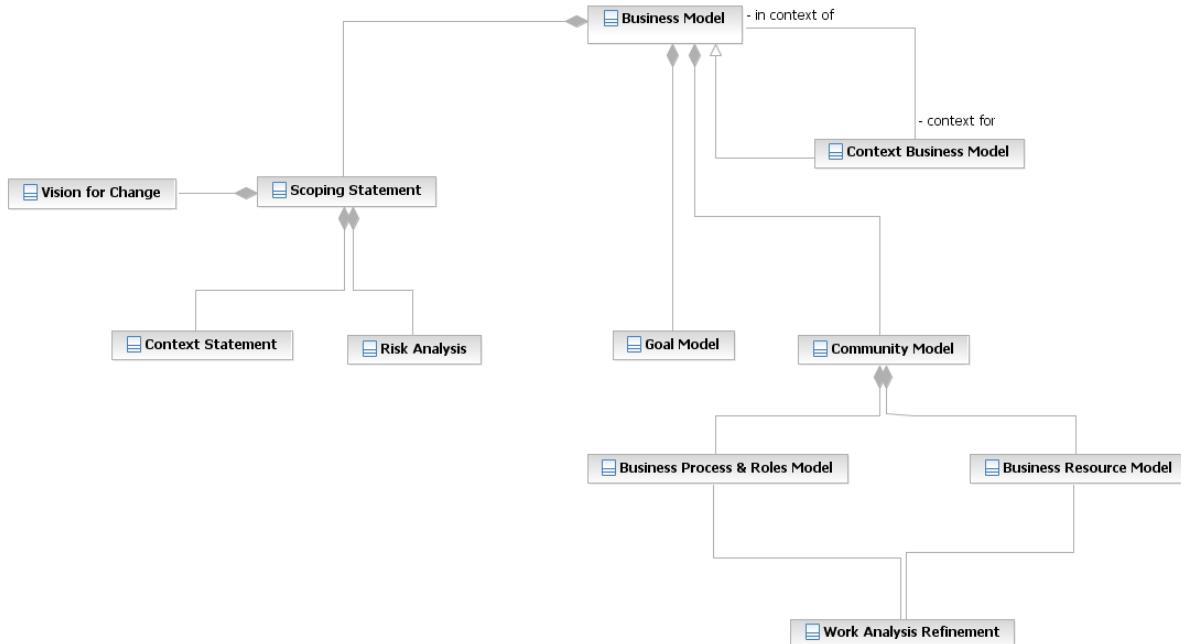


Figure 17: COMET Framework for Business Modelling. Adapted from [53].

¹⁰ <http://modelbased.net/comet/>

¹¹ <http://www.sintef.no/Home/>

¹² Business Motivation Model: <http://www.omg.org/cgi-bin/doc?dtc/2006-08-03>

COMET has three different modelling domains which are mainly based on UML: Business modelling (**Figure 17**), requirement modelling and architecture modelling. Services are identified in a Work analysis Refinement (WARM) in the business model. Although the framework calls these services *business services*, they only represent services that are delivered by IT. An identified business service can be mapped to a *business service component* in the architecture model.

Enterprise Modelling Language¹³ (EML)

EML is a notation developed to enhance visual representations of business processes. It is developed by Richard Li at the University of Auckland [49] and is currently work in progress. Tool support has been implemented in Marama Metatools [54].

In EML, services are modelled first. Then operations are defined under each service. Operations play the same role as tasks in business process modelling and can be orchestrated by a process overlay.

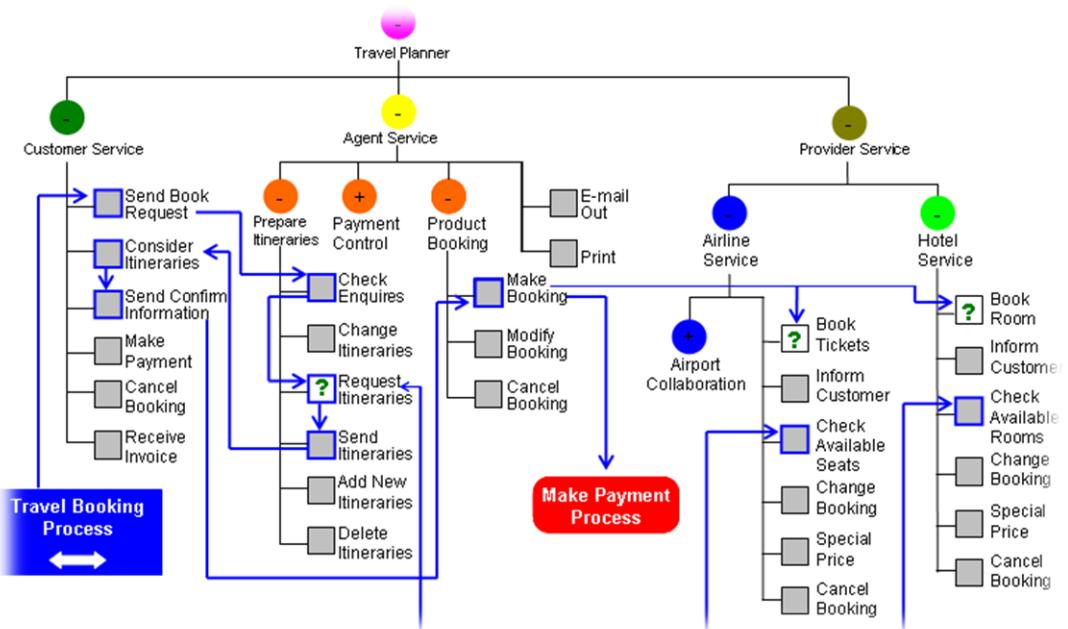


Figure 18: EML Process Overlay [49].

EML has no mechanism for service identification.

Service- Oriented Architecture and Modelling (SOMA)

SOMA is a well established methodology for SOA modelling developed by IBM with tool support in Rational Software Modeler¹⁴. It describes techniques for modelling, analyzing and designing SOA solutions.

The first step in SOMA is service identification. Both a top-down and bottom-up approach is used which includes methods such as domain decomposition and analysis of existing systems. In addition, a meet-in-the-middle approach is used to identify services that haven't been captured from the two other approaches. The technique used for this is goal-service

¹³ <https://wiki.auckland.ac.nz/display/csidst/Enterprise+design+tool+including+BPMN+and+BPEL+generation>

¹⁴ Rational Software Modeler: <http://www-01.ibm.com/software/awdtools/modeler/swmodeler/>

modelling. **Figure 19** shows that services can be identified from different types of business models.

In a service specification step, the identified services are ranked. Services with the highest score are the ones considered for specification. Functional and technical requirements are other inputs to the specification process. Finally, a service specification is created in Web Services Description Language¹⁵ (WSDL).

SOMA is an iterative methodology. Specifications are refined in each iteration. In the final realization step architectural and design decisions are made.

In SOMA, service means software service which is realized as web service¹⁶. SOMA lacks the possibility of representing services in business modelling.

Different tools based on Eclipse¹⁷ technology are used depending on which abstraction level work is done. The degree of model-driven identification of services from business models and other artefacts depends on the realization of SOMA. However, since it does not support service specification in the business domain, it cannot support business process identification of business services.

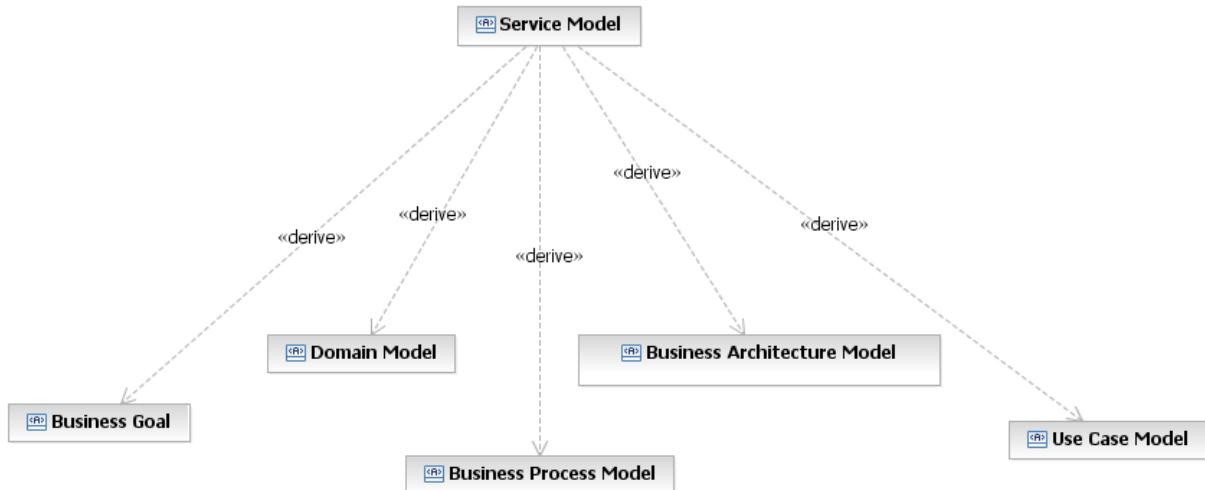


Figure 19: SOMA Service Identification. Adapted from [19].

Evaluation

The result of the evaluation of requirements is given in **Table 4**.

0: Requirement is not fulfilled

1: Requirement is partially fulfilled

2: Requirement is fulfilled

¹⁵ <http://www.w3.org/TR/wsdl>

¹⁶ <http://www.w3.org/2002/ws/>

¹⁷ <http://www.eclipse.org>

Requirement	SOMA	COMET	EML
TR1 Open Source Technology	1	2	2
TR2 Metamodel-based	2	2	2
TR3 Abstract vs. Concrete Syntax	2	2	1
TR4 Service Behaviour	2	1	2
TR5 Automated Process-driven Service Identification	0	0	0
TR6 Selective Abstractions	0	0	2
TR7 Service Traceability	1	0	0
TR8 Separation of Context- Viewpoints	0	0	0
TR9 Model-transformation	2	2	0
Sum	10	9	9

Table 4: *State-of-the-Art Evaluation.*

4.4. Summary

In this chapter, requirements for the solution were described. An evaluation of these will be used to assess the solution in Chapter 8. Finally, an evaluation of state-of-the-art tool frameworks was carried out with regards to the tool requirements defined in section 4.2. The PROSERVE tool will be compared to these tools in section 8.4.

Chapter 5. Theoretical Solution

In this chapter, a metamodel is proposed with the purpose of providing a solution for the research needs and the PROSERVE tool. The metamodel is specified in section 5.1 which includes a definition of a service context scheme in section 5.1.3. A solution for process-driven business service identification is proposed in section 5.2. Finally, section 5.3 depicts a service architecture that serves to align different artefacts.

This theoretical solution consists of two parts: (a) the PROSERVE metamodel and (b) model transformation rules for business processes. While the PROSERVE metamodel provides a solution for context-views and context-interaction, the model transformation rules provide a foundation for process-driven business service identification. Solution part b) is dependent on solution part a). Notation and a mechanism for context-overview are outlined in Chapter 6.

A sub-set of this solution will be realized in a proof-of-concept tool. Implementation of the tool is described in Chapter 6.

5.1. PROSERVE Metamodel

The main purpose of a metamodel is to create a solution for context-views and context-interaction. In order to separate different contexts, a taxonomy for services needs to be created. Based on this taxonomy, a service context scheme is proposed as a solution for context support. The solution leverages on UPMS and OSOAR concepts.

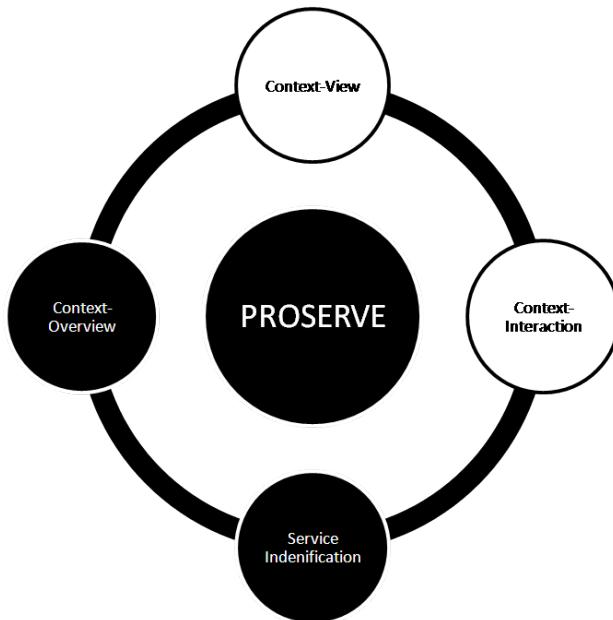


Figure 20: Resolved Concepts in the Metamodel¹⁸.

5.1.1. Taxonomy

A classification of service definitions is required to clarify semantics and the contexts they operate in. The reviewed literature reveals that the term *service* has different meanings depending on the context in which it is being used. In business, and especially in marketing,

¹⁸ The concepts were explained in section 3.3.

services are custom-oriented as described by Vargo [6]. In software, on the other hand, service is a metaphor for software components [55].

In this work, the service term is differentiated in order to provide a taxonomy that clarifies the environment (domain) it exists in and its type of interaction. Definitions of different types of services are here derived from Vargo's service definition [6]:

- *E-service (Electronic Service): The application of specialized competences (knowledge and skills) through deeds, processes and performances for the benefit of an entity or the entity itself, mediated through human interaction with information technology across the boundary of the enterprise.*
- *H-service (Human Service): The application of specialized competences (knowledge and skills) through deeds, processes and performances for the benefit of an entity or the entity itself, mediated through human interaction across the boundary of the enterprise.*
- *EE-service (Enterprise Electronic Service): The application of specialized competences (knowledge and skills) through deeds, processes and performances for the benefit of an entity or the entity itself, mediated through information technology within the enterprise.*
- *EH-service (Enterprise Human Service): The application of specialized competences (knowledge and skills) through deeds, processes and performances for the benefit of an entity or the entity itself, mediated through human labour within the enterprise.*
- *S-service (Software Service): The application of specialized competences (knowledge and skills) through software components and performances for the benefit of a software entity or the software entity itself.*

The notion of *business service* is regarded as a generalization of all these services except from S-service.

5.1.2. Core

The purpose of this section is to outline the model core for PROSERVE metamodel, which the proposed service context scheme is built on. This model foundation is based on UPMS and OSOAR concepts. The entities are described in details in section [5.1.6](#).

Participants interact with other participants through the notion of *service* and *requisition*. A participant can offer one or multiple services which can be consumed through requisitions. Moreover, a participant can also consume its own services.

A relationship between a service and a requisition requires an agreement of a *service contract*. A service contract can be specialized into two types of contracts. First, a *service fulfilment contract* can define how a service must be realized through definitions of *operations*. Secondly, a *service delivery contract* can describe what the service owner have to deliver and what the requisition owner have to deliver in order to receive the delivery. *Service operations* are used to define specific deliveries.

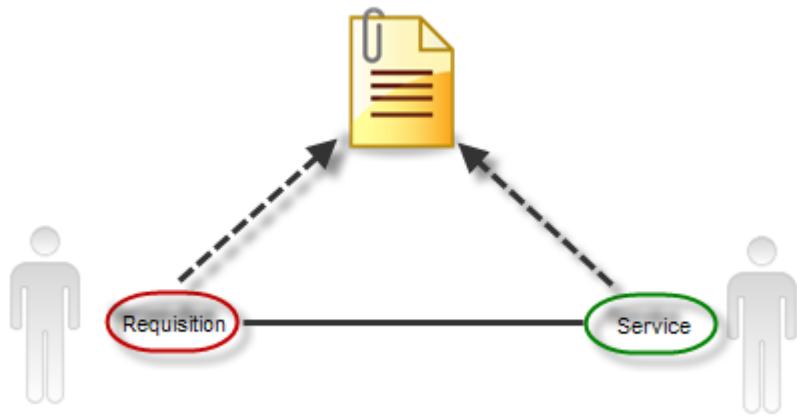


Figure 21: Service Contract Agreement.

In *service delegation*, a part of a service delivery is delegated to other services than the one responsible for the total delivery in a given service contract. When a service delegation is made, a new requisition is created. The requisition is then associated with a service contract. In order to minimize coupling to participants, service delegation is done directly from the *service* that needs to delegate. Service delegation is called *EnterpriseServiceDelegation* in PROSERVE.

A *branch* is used to partition services and requisitions in order to create a better overview. Branches are specific to whether they support service aggregation or requisition aggregation.

In order to support the notion of service behaviour, operations and requisitions are enabled to point at its successors. A successor is either another operation or another requisition. A requisition is used where an operation needs to be followed by a service. By using a requisition rather than a service to represent the activity, the behaviour is decoupled from specific services.

5.1.3. Context Scheme

In order to be able to clarify the context of a service, the idea of *participant types* is used. These are specified along two axes: nature of the entity and context-view. While the nature of an entity is either human or electronic, a context-view is a *business view*, *integration view* or a *software view*. A human entity is either a person or a group of people, for instance people in an organizational unit. Electronic entities, on the other hand, can represent software applications, software modules or software specifications. Views are explained below. Finally, an interaction denotes a service-requisition relationship between two participants.

Participants represent entities in a service network that have capabilities expressed through services and needs through requisitions. To support context, these are specialized into the following participant types:

- *Collaborator*
- *EnterpriseSoftwareParticipant*
- *EnterpriseHumanParticipant*
- *SoftwareParticipant*

A *Collaborator* is defined as an external entity outside an enterprise. It can be either electronic or human. It can also be an external entity in a business-to-business relationship. A *Collaborator* belongs to a business view.

EnterpriseHumanParticipants and *EnterpriseSoftwareParticipants*, collectively called *enterprise components*, are internal entities within an organization. While an *EnterpriseHumanParticipant* represents a human entity, an *EnterpriseSoftwareParticipant* represents an electronic entity. They both belong to a business view and can interact with each other and *Collaborators*.

A *SoftwareParticipant* is a participant that represents a software entity, in particular a software specification. It belongs to a software view and can interact with other *SoftwareParticipants*.

In order to enable interaction between a business view and a software view, an integration view is constructed for coupling of requisitions from *EnterpriseSoftwareParticipants* with services owned by *SoftwareParticipants*.

A context for a service is created by assigning participant ownerships for it. A service context scheme outlined in **Figure 22** defines contexts for interactions based on these ownerships.

The legend is enclosed in an orange rounded rectangle and contains four items:

- Interaction in Business Domain (blue square)
- Interaction in Integration Domain (green square)
- Interaction in Software Domain (yellow square)
- Illegal Interaction (black square)

		Enterprise Software	Enterprise Human	Software
		Electronic	Human	Electronic
Collaborator	Electronic	E-service	E-service	
	Human	E-service	H-service	
Enterprise Software	Electronic	EE-service	EE-service	S-service
Enterprise Human	Human	EE-service	EH-service	
Software	Electronic	S-service		S-service

Figure 22: Service Context Scheme.

In the context scheme in **Figure 22**, rules for extracting context-views and context-interaction are defined. Rules for interaction apply both within a view and across the views. Illegal interactions are also stated.

Business realization is a result of service networking where participants interact through different contexts. **Figure 23** illustrates how participants can interact in such a network.

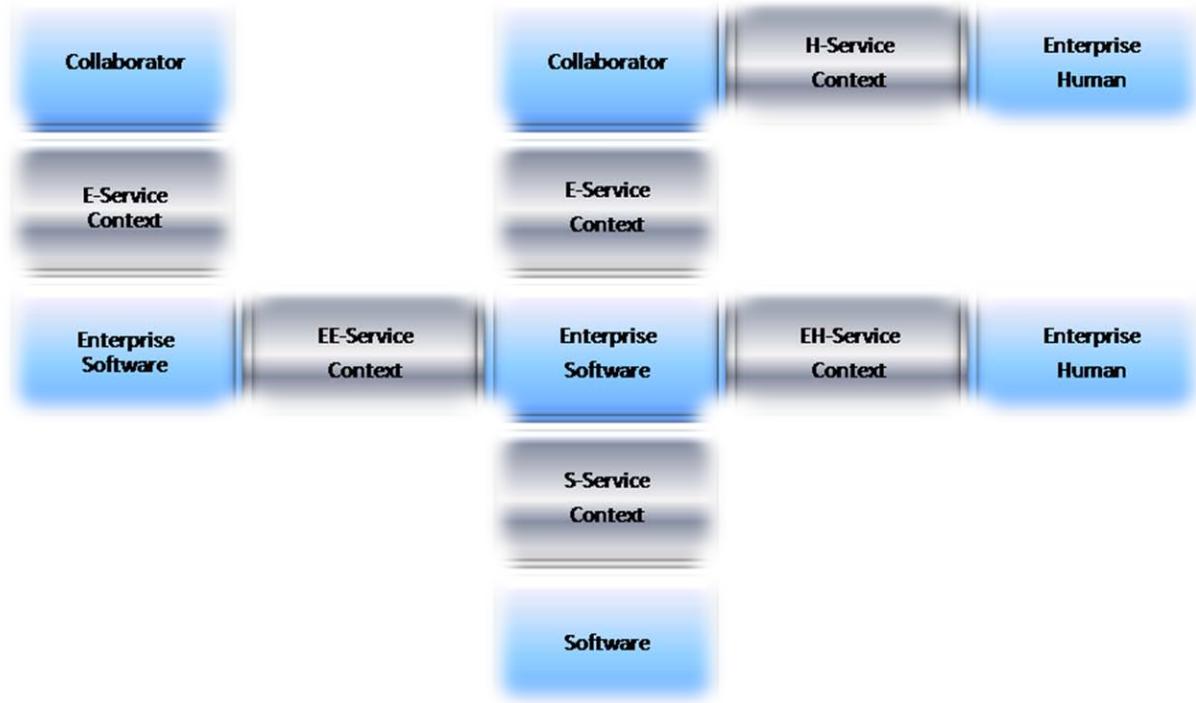


Figure 23: Service Network.

5.1.4. Service Splitting

Service splitting¹⁹ is created as a means to depict that one service request can invoke service delivery to multiple participants.

When a service is requested that will result in service delivery to another entity, a separate service for this entity must be made. The two services can then be joined in a common service. In order to explain this better, an example is given.

In the case study, a *clinical department* requests a service that an SMS is sent to a patient to remind him or her about an appointment. In this way, the service also fulfills a *patient* requisition to get reminded about the appointment. Based on the concept of service splitting, these two requisitions have to be fulfilled by different services.

¹⁹ The notion of service splitting supports Vargo's focus on co-production of value.

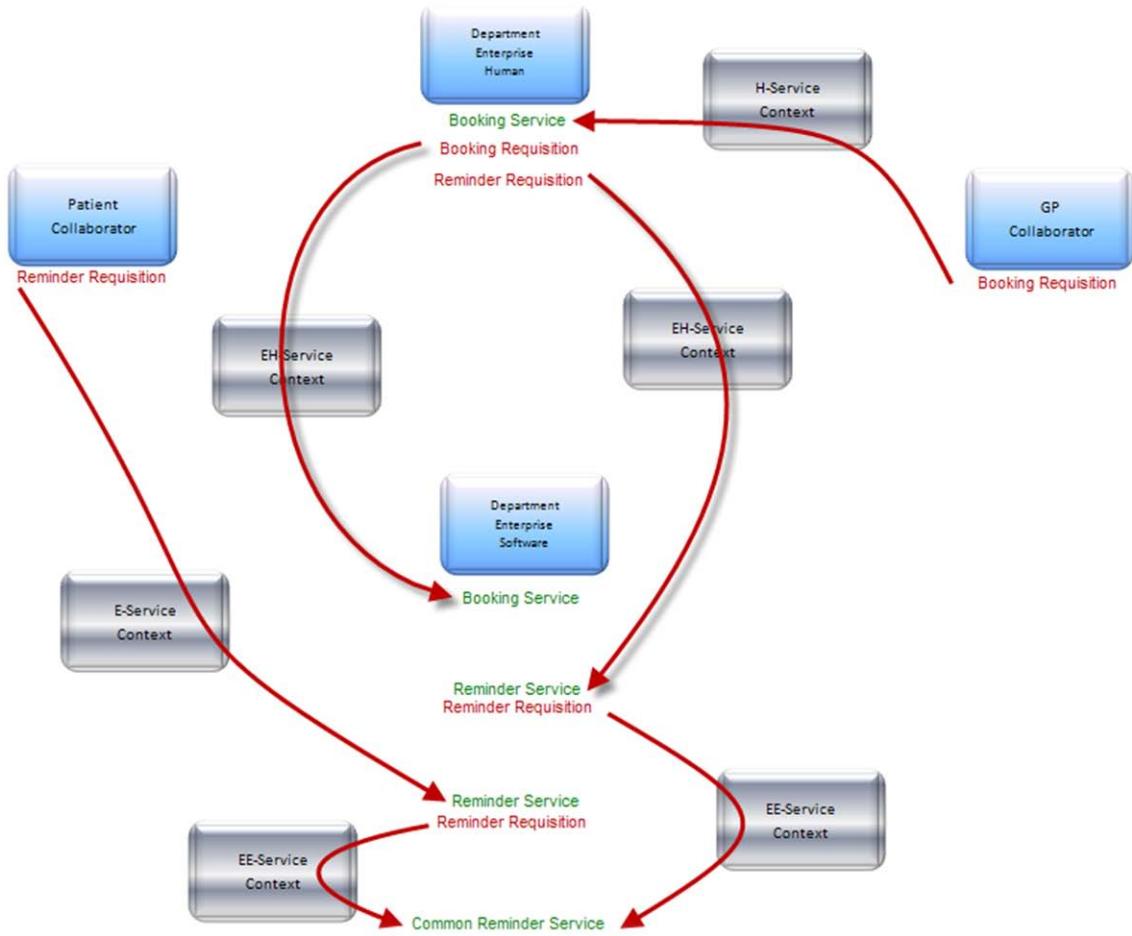


Figure 24: Service Splitting.

In **Figure 24**, two different reminder services are defined in order to depict deliveries to different types of participants. These two services are then joined in a common service.

5.1.5. Views

PROSERVE specifies three different views on its metamodel. A business view for enterprise architects and a software view for software architects. Between these views is an integration view, where an enterprise architect and a software architect can couple business requisitions to software services in a forward-driven way.

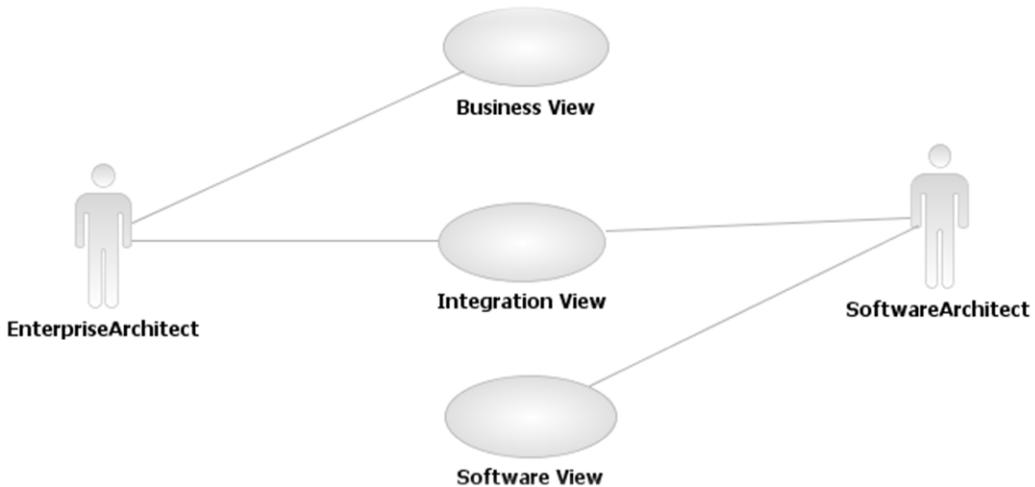


Figure 25: Use Case of PROSERVE.

Views are abstracted based on the rules defined in the service context scheme (**Table 5**).

A *business view* can abstract business relevant information from the model in order to provide a focused view on business concerns. This view will hold representations of services that hide couplings to IT.

On the other side of the gap is the *software view*. This view represents the concern of software services that need software service specifications. Although this view hides coupling to representations of business services, the underlying model captures traces to services in the business view.

To bridge the gap between the business view and the software view, an *integration view* is defined. This view has the responsibility of reconciliation between business and IT. It can represent requisition from the business view and services that belong to the software view. **Figure 26** depicts different views on the common model. Services, requisitions and participant types are context sensitive artefacts. In order to abstract services and requisitions to the right view, their ownerships are queried. The participant type of their owner determines to which view they belong, as given by the service context scheme.

View	Context	Participant
Business	H-service, E-service, EE-service, EH-service	Collaborator, EnterpriseSoftware, EnterpriseHuman ²⁰
Integration	EH-service, S-service	EnterpriseSoftware, Software
Software	S-service	Software

Table 5: Abstraction of Views Based on Participant Types.

²⁰ Ownerless services and requisitions are also part of the business view.

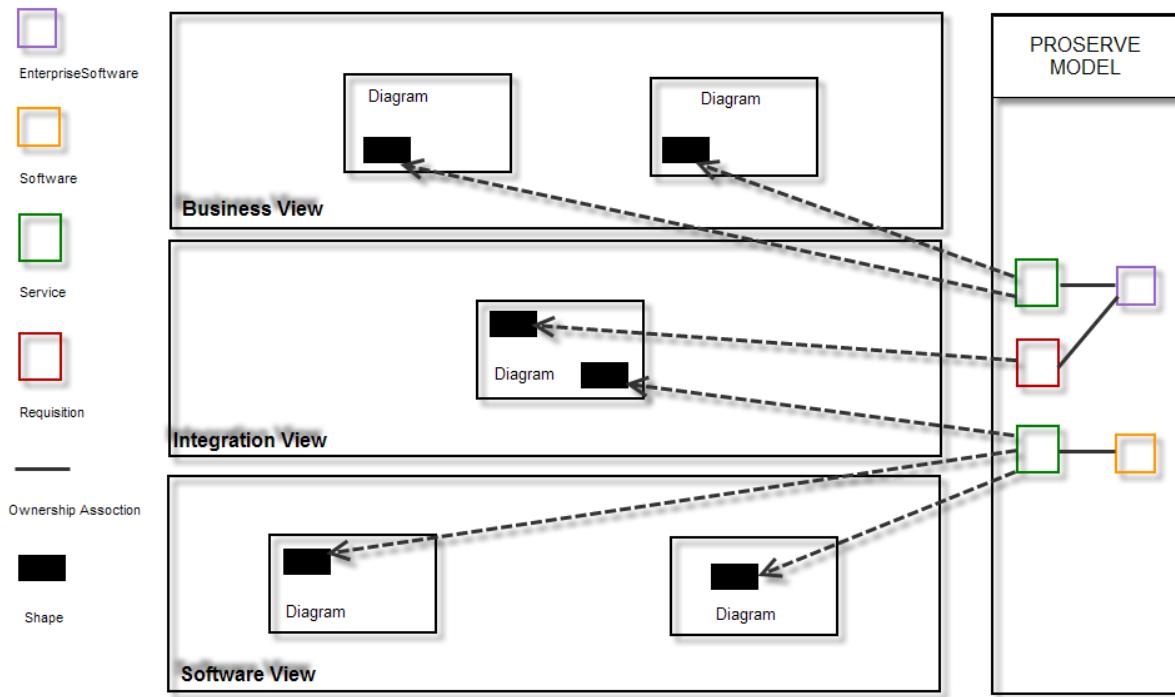


Figure 26: Views on the PROSERVE Model.

The means to facilitate interaction between the views is *service delegation*²¹. When it is decided to outsource a business service to IT, it is assigned to an *EnterpriseSoftwareParticipant* in the business view. A new delegation is then made to request a service that belongs in the software view. The requisition made, can then be abstracted from the model to the integration view. After a service contract is specified, a new service can be derived. The new service is assigned to a *SoftwareParticipant*. It can then be abstracted to the software view. **Figure 27** illustrates how service delegation is used between the different participant types.

²¹ EnterpriseServiceDelegation

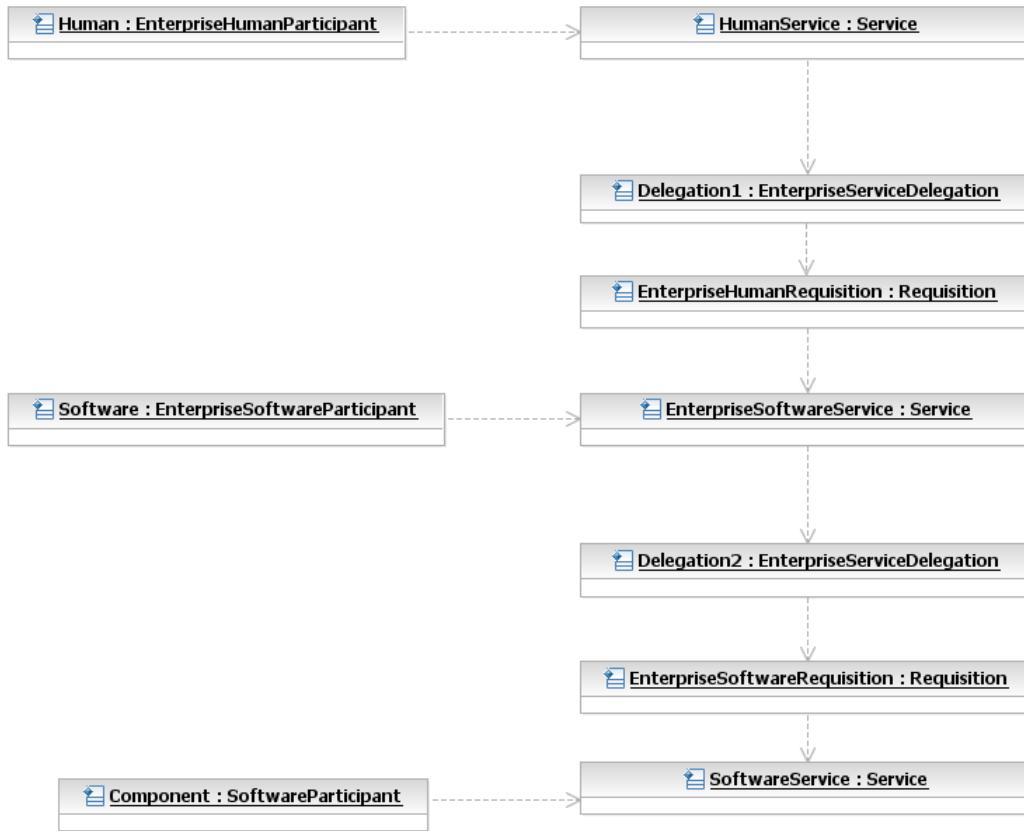


Figure 27: Service Delegation.

In each view, a service may be recursively decomposed in order to create finer-grained services to distribute responsibilities and facilitate re-use.

5.1.6. Metamodel

Details of the entities in the PROSERVE metamodel are described in this section.

Branch

A branch can aggregate services and requisitions. It is used for partitioning, so that a better overview can be provided.

Associations: Association to Service and Requisition.

Constraint: Services and requisitions cannot be added to the same branch. It is either a service branch or a requisition branch.

Collaborator

A *Collaborator* represents an entity outside an enterprise.

Generalization: Participant.

Constraints: Cannot connect to services owned by SoftwareParticipants.

Views: Business.

EnterpriseHumanParticipant (EHP)

An *EnterpriseHumanParticipant* (EHP) represents a human resource within an enterprise. If a service is owned by an EHP, it means that the service is provided by human effort.

Generalization: Participants.

Constraints: Cannot connect to services or requisitions owned by SoftwareParticipants.

Views: Business.

EnterpriseServiceDelegation

An *EnterpriseServiceDelegation* is a mechanism for expressing that a service is dependent on other services.

Associations: Associations to Requisition and Service.

EnterpriseSoftwareParticipant (ESP)

An *EnterpriseSoftwareParticipant* (ESP) represents applications within an enterprise. However, these participants do not represent the software itself. They conceptually represent business that is outsourced to software. Services that are owned by ESP are decoupled from specific software technologies and software design. ESP and EHP can represent complementary parts of organizational units.

Generalization: Participants.

Views: Business and Integration.

Operation

An *Operation* describes an activity that has to be performed in order to deliver a service. It can capture its successor in a chain of sequential activities. A successor is either another *Operation* or a requisition. An operation may have multiple successors.

Associations: Association to ServiceFulfilmentContract.

Participant

A *Participant* represents an entity which can own services or requisitions. It is responsible for complete service delivery of the services that it owns. Realization of the service can be done exclusively by the *Participant* or it can delegate parts and even the whole delivery to other services. *Participants* can be linked to organizational structures.

Associations: Associations to Service and Requisition.

Requisition

A *Requisition* is a container label for service deliveries that are required. It collectively represents the required service deliveries and is a point of communication to services.

Associations: ServiceContract and ServiceDelegation.

Service

A *Service* is a container label for service deliveries that are offered. It collectively represents the provided service deliveries and is a point of communication to requisitions.

Associations: ServiceContract and ServiceDelegation.

ServiceContract

A *ServiceContract* is an agreement between a requisition and a service. An association between a service and a requisition is an acceptance of the contract.

Associations: Associations to *Service*, *Requisition* and *ServiceRole*.

ServiceDeliveryContract

A *ServiceDeliveryContract* specifies what the service has to deliver. It contains *ServiceOperations*. By specifying multiple *ServiceOperations* in the contract, different deliveries can be specified in the same contract.

Generalization: *ServiceContract*.

Associations: *ServiceOperations*.

ServiceFulfilmentContract

A *ServiceFulfilmentContract* describes how a service must be realized. It contains specifications of operations that represent activities that have to be performed by a participant that provides the service.

Generalization: *ServiceContract*.

Associations: *Operations*.

ServiceOperation

A *ServiceOperation* describes what the service actually delivers. It can also describe inputs needed to produce delivery. Multiple service deliveries can be specified in one *ServiceDeliveryContract*.

Associations: Association to *ServiceDeliveryContract*.

ServiceRole

A *ServiceRole* holds traceability information. It keeps track of a *ServiceContract*'s corresponding sub-process and its parent in a business process model.

Associations: Association to *ServiceContract*.

SoftwareParticipant (SP)

A *SoftwareParticipant* (SP) represents a software entity, e.g. software specifications. They can fulfil requisitions made by ESPs.

Generalization: *Participants*.

Constraints: Cannot connect to services or requisitions owned by Collaborator or EHP.

Views: *Software*.

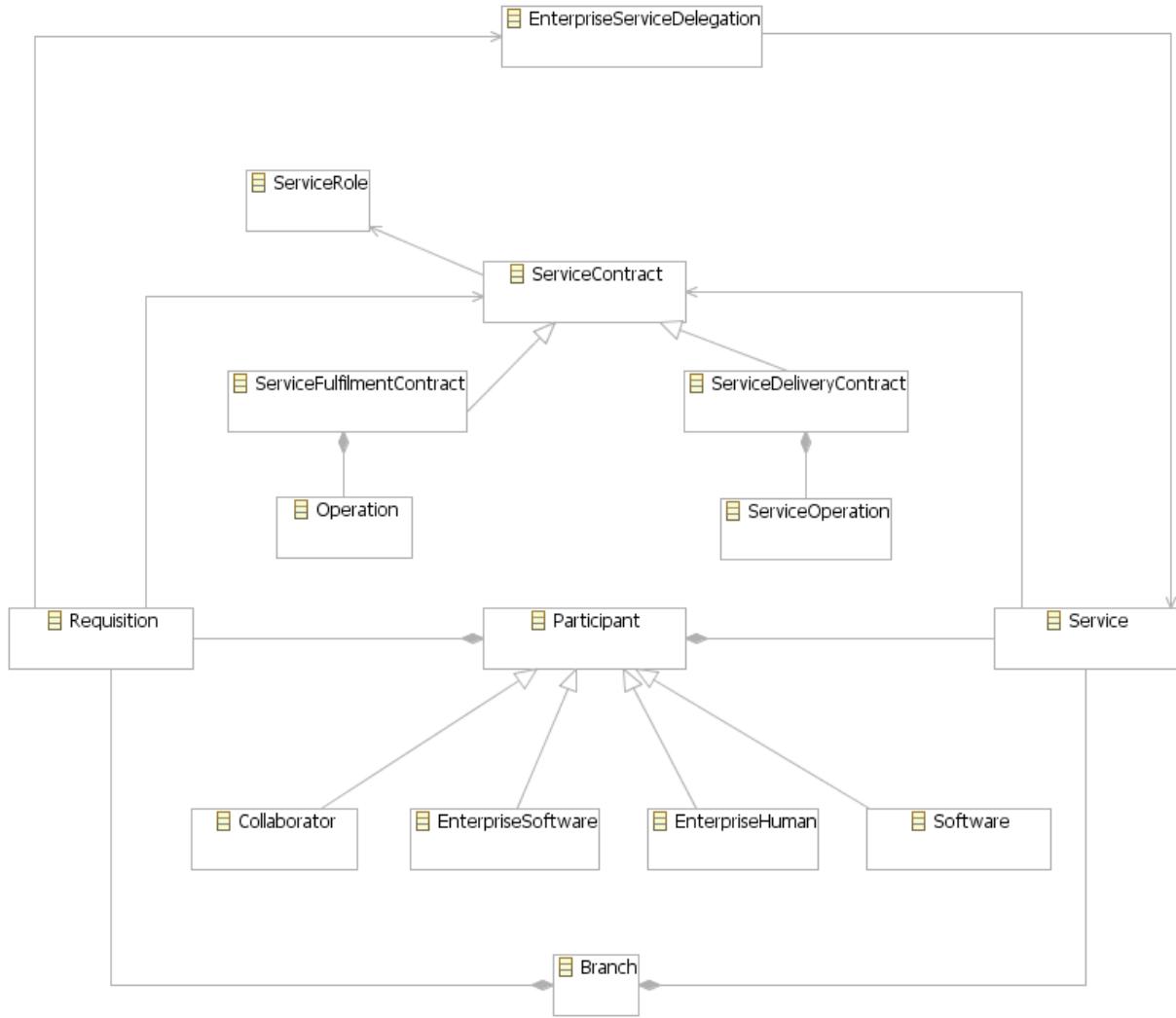


Figure 28: PROSERVE Metamodel.

5.1.7. Limitations

This metamodel only considers forward engineering of services. It has no support for merging new services with existing ones in a credible way. In the real world, however, existing services must be able to couple with other services or requisitions.

Moreover, some features are excluded such as messaging between consumer and provider. Such a feature is included in frameworks in UPMS and SOMA and could conceptually be realized in a similar way in PROSERVE. The limitations are delegated to future work.

5.2. Process-driven Business Service Identification

The purpose of this section is to propose rules for how PROSERVE artefacts can be identified from business process models. BPMN is chosen as a source metamodel for identification of business services. The goal is to create transformation rules that can be used for automatic identification of these services.

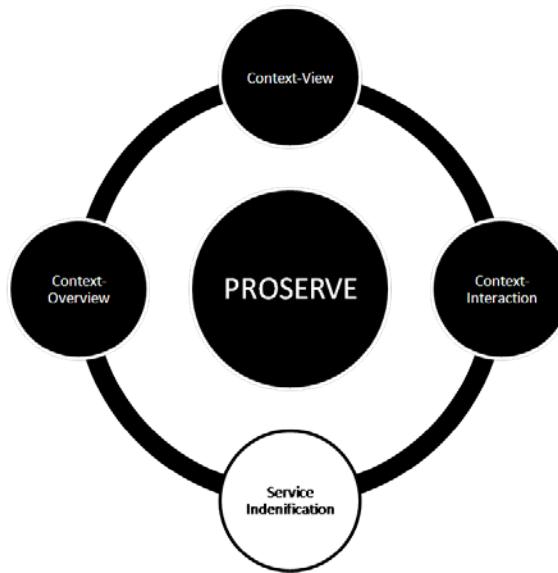


Figure 29: Resolved Concept.

Identifying Services from BPMN

A process encompasses process artefacts. Having the property of being a container for artefacts, it is mapped to the service entity in PROSERVE. The top-level process is, however, not represented as an entity in BPMN. The *BPMNDiagram* is chosen to represent this process, since it is a root entity for artefacts in the process. A sub-process is, on the other hand, represented as an entity. The following mapping is then proposed:

Rule 1: *BPMNDiagram* -> *Service*

Rule 2: *BPMNSub-process* -> *Service*

Identifying Requisitions from BPMN

Requisitions are counterparts of services. They are therefore also derived from Sub-processes.

Rule 3: *BPMNSub-process* -> *Requisition*

Identifying Service Operations and Operations from BPMN

Task activities describe sequentially *how* a business process is realized. It is therefore mapped to *Operations* in a *ServiceFulfilmentContract*.

ServiceOperations, on the other hand, need to capture the resulting service delivery. However, there is no entity in BPMN that captures this in an unambiguous way. No mapping is therefore suggested.

Rule 4: *BPMNTasks* -> *Operations*²²

Identifying EnterpriseServiceDelegations from BPMN

²² Tasks in the top-level process will not be transformed, since they are not encompassed by a sub-process.

Delegations occur when a BPMN model has defined sub-processes. Sub-processes support the process they are defined from. As a consequence, a service delegation is derived for every sub-process in the diagram:

Rule 5: *BPMNSub-process -> EnterpriseServiceDelegation*

Identifying ServiceFulfilmentContracts and ServiceDeliveryContracts from BPMN

ServiceFulfilmentContracts and *ServiceDeliveryContracts* are agreements of service fulfilment and service delivery between a requisition and a service. They are derived from the sub-process from which their related service originates.

Rule 6: *BPMNSub-process -> ServiceFulfilmentContract*

Rule 7: *BPMNSub-process -> ServiceDeliveryContract*

Identifying Service Role from BPMN

A *ServiceRole* holds traceability information. The following information is passed on:

Rule 8: *BPMNSub-process -> ServiceRole*

Rule 9: *BPMNSub-process id -> ServiceRole attribute: BPMNid*

Rule 10: *BPMNSub-process parent-id -> ServiceRole attribute: BPMNparentId*

Identifying Participants from BPMN

Participants may be identified from lanes and pools in BPMN. However, it is suggested that participants, lanes and pools are reconciled with an organizational model. Harmonization of business process models, service models and organizational models is addressed as future work.

Figure 30 shows how a sub-process and its tasks are mapped to PROSERVE. Red lines denote mappings with related semantics, while green lines show mappings less related semantics.

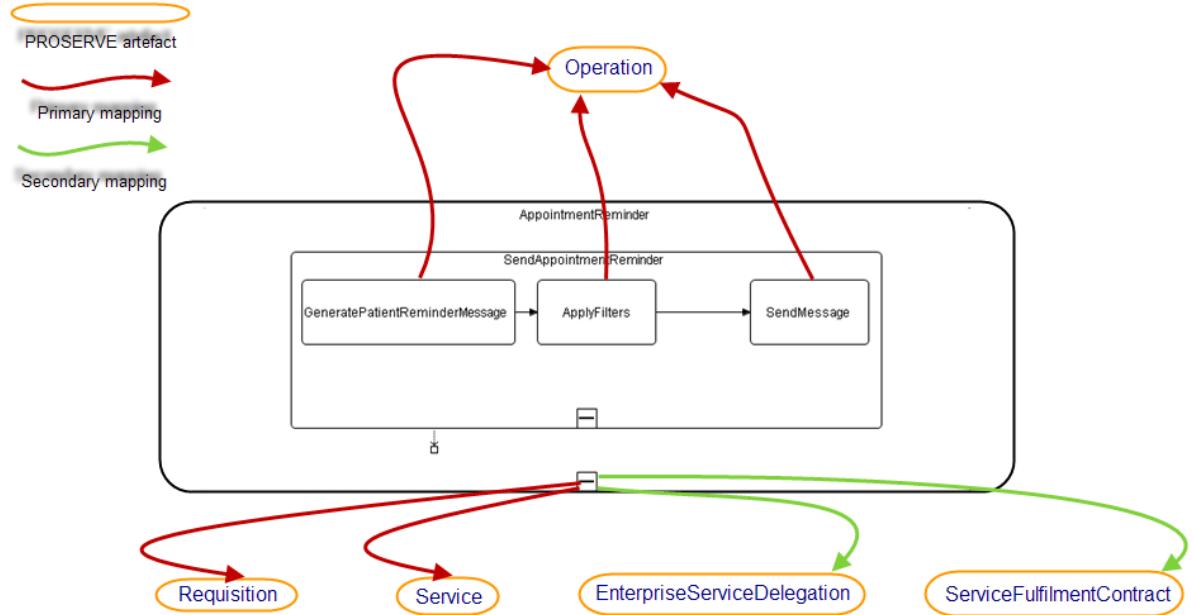


Figure 30: Derivation of PROSERVE Artefacts from BPMN.

5.2.1. Behaviour

The concept of activities having a specific ordering in order to fulfil a service is called service behaviour²³. For PROSERVE, a simple approach is taken: the derived behaviour capture the order of sub-processes and tasks. Elements used in service behaviour are operations and requisitions, respectively. The mechanism used to trace the order, is a mapping of the successor of a BPMN activity to a *successor* attribute in the derived artefact. In cases where sub-processes and tasks are followed by a gateway, multiple successors can be mapped. The type of gateway itself is not mapped, i.e. only forking and merging of activities are depicted in service behaviour.

5.3. Service Architecture

The purpose of outlining an architecture²⁴ here is to align PROSERVE artefacts.

The architecture for services encompasses both business and software concerns. As a consequence an effort is made to create an interdisciplinary architecture that includes both domains.

Business architecture provides different views of significant parts of the business [19]. In this framework, a business service layer is used. It is suggested that an organizational view may be linked to enterprise components in this layer. Other views may also be used in order to create a complete business architecture.

Software architecture on the other hand denotes the structure or structures of a system, which comprise software components [56]. Software architecture is represented as a software service layer (**Figure 31**).

²³ UPMS concept

²⁴ Inspired by IBM Solution Stack: <http://www-128.ibm.com/developerworks/library/ar-archtemp/index.html>



Figure 31: Service Architecture.

The top layer in the architecture is *collaborators* in order to focus on the concept of co-production of value between the business and the customers [6] or partners. The *collaborators* layer is a container for services and requisitions owned by collaborators.

Processes exist in both in business service layer and software service layer as business processes and services behaviours, respectively.

Finally, the *IT-infrastructure* layer is an abstraction over existing software applications and data repositories.

5.4. Summary

In this chapter, a solution for how to provide context-views and context-interaction through a service context scheme was proposed. Moreover, a solution for identifying business services from business process models was created. Finally, the solution was wrapped into an architecture for services. The definition of the PROSERVE metamodel will be implemented in a tool for proof-of-concepts in Chapter 6.

Chapter 6. The PROSERVE Tool

This chapter explains how a technical solution was made in order to prove concepts for the theoretical solution proposed in Chapter 5. First, section 6.1 describes the concepts to be proven. Section 6.2 outlines the selected technology used to implement the solution. The PROSERVE notation is described in section 6.3, while section 6.4 explains the diagrams. Layout design of service trees is included here. Section 6.5.2 provides an overview over the components of the system. Finally, section 6.6 explains how model transformation and service trees were implemented in the PROSERVE tool.

6.1. Subset of Problem to be Proven

The goal of this prototype is to provide a proof-of-concept for the PROSERVE metamodel and the notion of service trees. There are proposed solutions in the metamodel that need to be proven:

- a. Service context scheme
- b. Process-driven business service identification

It needs to be proven that concept a) provides a solution for context-views and context-interaction.

Due to a shortage of time, only a partial proof for service context scheme as a context-interaction mechanism is proven. Moreover, *ServiceDeliveryContract* and *ServiceOperations* are not realized for the same reason.

The concept of service trees is implemented in the tool in order to provide a solution for context-overview.

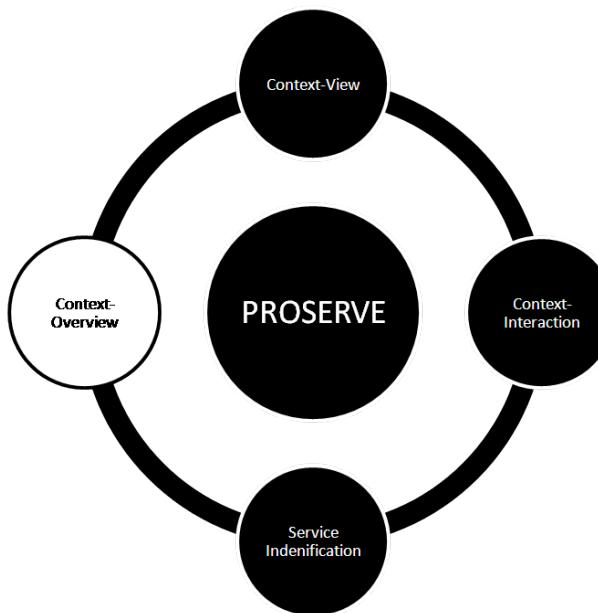


Figure 32: Resolved Concept.

6.2. Tools and Technologies

For rapid realization of the proof-of-concept, the prototype is based on existing tool frameworks. Two types of framework have been identified to support development of a graphical model editor with model to model propagation of data: A meta-tool framework and a model transformation framework.

6.2.1. Meta-tools

Meta-tools are defined as sets of software tools that support rapid specification and implementation of other software tools [54]. A wide range of meta-tools are available, but the time frame for this thesis work does not allow a thorough evaluation of all of these tools. Two Eclipse-based meta-tools are considered for the prototype: GMF [57] and Marama [54].

GMF (Graphical Modeling Framework) is based on the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF), and provides a framework for developing graphical editors [58, 59]. Metamodel definitions are based on EMF, which is a Java implementation of a core subset of OMG-MOF [12] called Ecore. GEF enables creation of graphical editors from an existing application model such as made in EMF. GMF's graphical definition models contain information related to graphical elements occurring in the GEF-based runtime environment [58].

Marama is a set of meta-tools developed at the University of Auckland for rapid specification of notational elements, meta-models, view editors and view-model mappings [54]. The tools leverage on the GEF and EMF as shown in **Figure 33**.

Criteria	GMF	Marama
Ecore support	2	0
Multiple level inheritance	2	0
Aggregation	2	0
Model transformation support integrated	0	2
Visual support for notation development	0	2
Visual icon-entity mapper	0	2
Event-handlers	(1) Must be coded	(2) Event-handler templates for editor
Visual constrainer for model behaviour	0	2
Multiple view on the same model	2	2
Sum	9	12

Table 6: Evaluation of Features

An evaluation of features (**Table 6**) suggests that Marama enables a quicker development of the prototype since it offers visual editors for developing the tool. The need for event-handlers and Marama's capability to rapidly generate event-handler *hooks* or templates make Marama the preferred choice.

Marama's main weakness in this setting is its metamodel editor which lacks features for multiple level inheritance and (UML) aggregation. In addition, it would be preferred to have a metamodel-tool that is based on Ecore to better align this work with UPMS which is a MOF

compliant standard. However, these concerns are not considered to be important for the proof-of-concept objectives. MOF support is planned in a future release of Marama²⁵

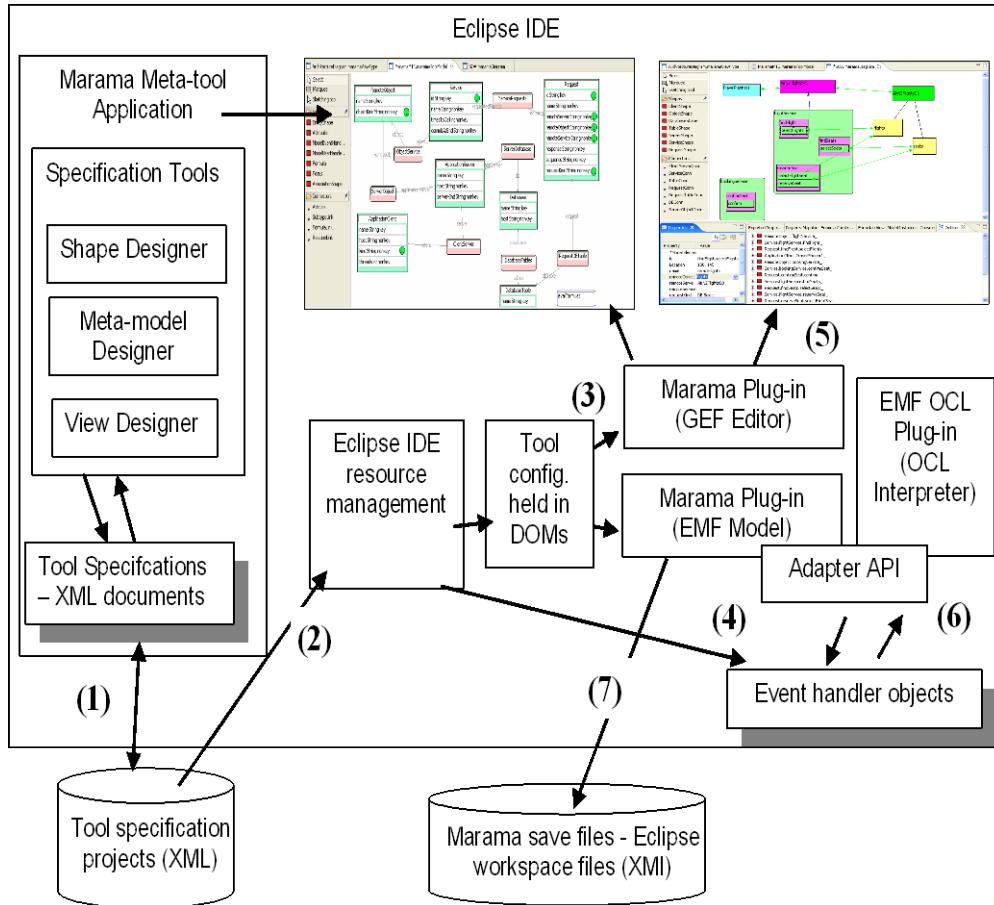


Figure 33: Marama Architecture [54].

In order to save additional time, an existing Eclipse modelling tool for BPMN is used. SOA Tools BPMN Modeler [60] is a graphical editor for creating BPMN diagrams based on GMF which uses an EMF model.

6.2.2. Model Transformation Framework

A model-to-model transformation is assessed to be a better strategy for modelling transformation than model-to-text, with regards to maintainability and scalability [61]. However, there are two reasons for choosing the latter implementation here. Firstly, Marama defined metamodels do not conform to any technology standard like MOF/ Ecore, which complicates a straight-forward approach using model-to-model transformation tools like ATL [62]. Secondly, a one-directional transformation is sufficient to support the described proof-of-concept objectives. Model transformation maintainability and scalability are not evaluated in this thesis.

Two tools are considered for model transformation: MaramaTorua [54] and MOFScript [63]. MaramaTorua is integrated with Marama and is based on XSLT²⁶, which is a language for transforming XML documents into other XML documents [64]. MOFScript, on the other

²⁵ Source: Prof. John Hosking at the University of Auckland

²⁶ XSL Transformations

hand, is a model-to-text transformation language and tool based on the QVT-Merge [65] specification that supports MOF model-to-text transformation [66]. A requirement for the model transformation with MOFScript is that the source model is expressed as an EMF/Ecore model [63].

The main reason for choosing MOFScript as the transformation tool for the prototype it is assessed to be more flexible with regards to implementing the transformation rules. However, the MaramaTorua is also assessed to be able to do the job.

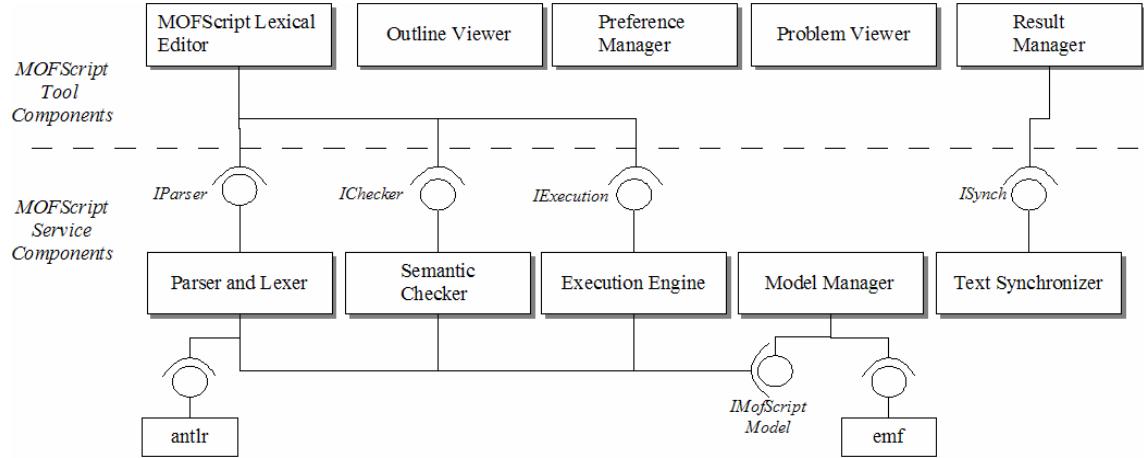


Figure 34: MOFScript Tool Architecture [66].

6.3. Notation

The purpose of this section is to describe the notation for PROSERVE.

6.3.1. Shapes

The diagrams use two types of shapes: Icons and connectors. An icon is a visual representation of a model entity instance, while connectors visually represent associations between these instances. **Table 7** shows the different icons used. No particular effort has been put in the design of these.

There are two types of connectors: simple lines and tree lines. A simple line is a straight line between two instances, while a tree line is a line with a 90-degree bend between two icons in order to create the tree structure.

Entity Name	Shape
Service	
Requisition	
Collaborator	
Enterprise Human	
Enterprise Software	
Software	
ServiceFulfilmentContract	
Operation	
EnterpriseServiceDelegation	
ServiceBranch	
RequisitionBranch	

Table 7: Entity Shapes (Icons).

6.3.2. Trees

As described in section 3.6, collapsible hierarchical representations will be used for navigating and scaling information. A tree structure is chosen for this purpose. The tree shape consists of tree nodes which are shapes that represent entity instances. In addition, it consists of connectors between the tree nodes and a control shape.

A branch consists of a node and its children at its adjacent sub level. It has two possible notational states: extended or collapsed. In the extended state, all children in the branch are displayed, while in the collapsed state none are shown. The control shape is used to indicate that a branch can be collapsed or extended.

6.4. Diagrams

This section outlines diagrams in the PROSERVE tool. Details of these are described in Appendix III.

6.4.1. Business View

The business view displays artefacts in two diagrams that are necessary to view from a business viewpoint (Figure 35). Model instances that are owned by *EnterpriseSoftwareParticipants*, *EnterpriseHumanParticipants* and ownerless elements can be shown in these diagrams. Shapes are clustered in service trees in this view.

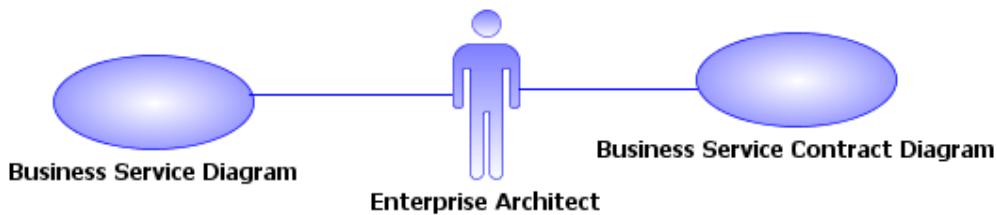


Figure 35: Diagrams in Business View.

The *Business Service Diagram* is the top level diagram in the business view. Its purpose is to provide an overview of service elements and to partition services and requisitions in branches.

A *Business Service Contract Diagram* enables coupling of a service and a requisition. It is also a diagram that can display service artefacts that are derived from a business process model.

6.4.2. Integrated View

The purpose of the integrated view is to enable coupling between the business and the software domain. Requisitions are imported from the business view and new services are derived for the software view. Shapes are clustered in service trees in this view.



Figure 36: Diagram in Integrated View.

A dedicated *Integration Service Contract Diagram* is made for coupling requisitions made by Enterprise Software to services owned by Software. Only requisitions that are owned by *EnterpriseSoftwareParticipants* are allowed to be imported for presentation in the diagram.

6.4.3. Software View

The software view contains a diagram where services can be dealt with from a software concern. Services that belong to this view can either be created both in this view and in the integration view. Only services and requisitions that are owned by *SoftwareParticipants* can be presented in this view. Shapes are clustered in service trees in this view.

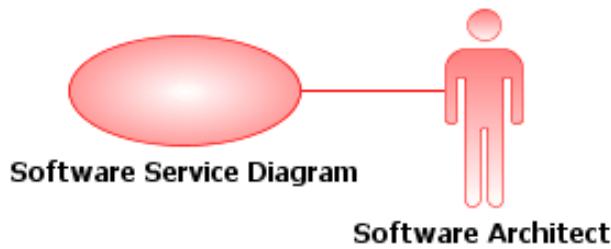


Figure 37: Diagram in Software View.

The *Software Service Diagram* provides an overview of service elements and a means to partition services and requisitions in branches.

6.4.4. All views

These diagrams can present artefacts from all views. While horizontal trees were used in the views above, diagrams that are view-independent use vertical trees.

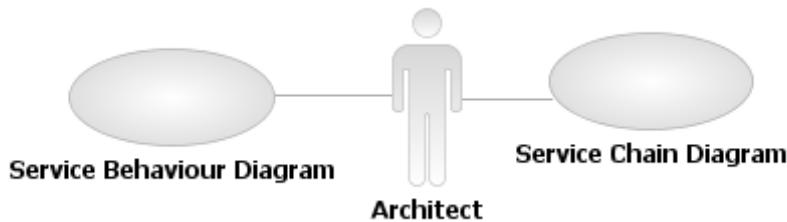


Figure 38: Domain independent diagrams.

A *Service Traceability Diagram* shows how a service is linked to other services through service-requisition relationships and service delegation. A service can be traced both toward its root service and along its chain of children.

Finally, a *Service Behaviour Diagram* shows the order of the operations in service interfaces to be executed.

6.4.5. Diagram Behaviour

A tree is constructed by dragging a node on top of another (drag-and-drop), so that they intersect. If both nodes have tree-node properties and they are allowed to connect, they form a tree structure.

If a service is dragged-and-dropped on top of a participant, an association is made between the participant and the service to represent that it is now owned by the participant. Moreover, this action causes a service branch to be attached to the participant and the service is attached to the service branch in the diagram (**Figure 39**).



Figure 39: Result of Shape Intersection between EnterpriseSoftware and Service.

The same action applies for requisitions, resulting in a RequisitionBranch being added to the tree (**Figure 40**).



Figure 40: Adding Requisition.

For each requisition or service dragged on to a participant, a new branch is attached. If a Requisition or a Service is dragged on a branch, they are attached to the branch.

A small grey square to the left of the branches, a control shape, indicate that the branch can be collapsed in order to hide the children of it. If the branch is in a collapsed state, it can be expanded to show its children (**Figure 41**). By right clicking a node with a control shape, an option for toggling collapsing or expansion is displayed.

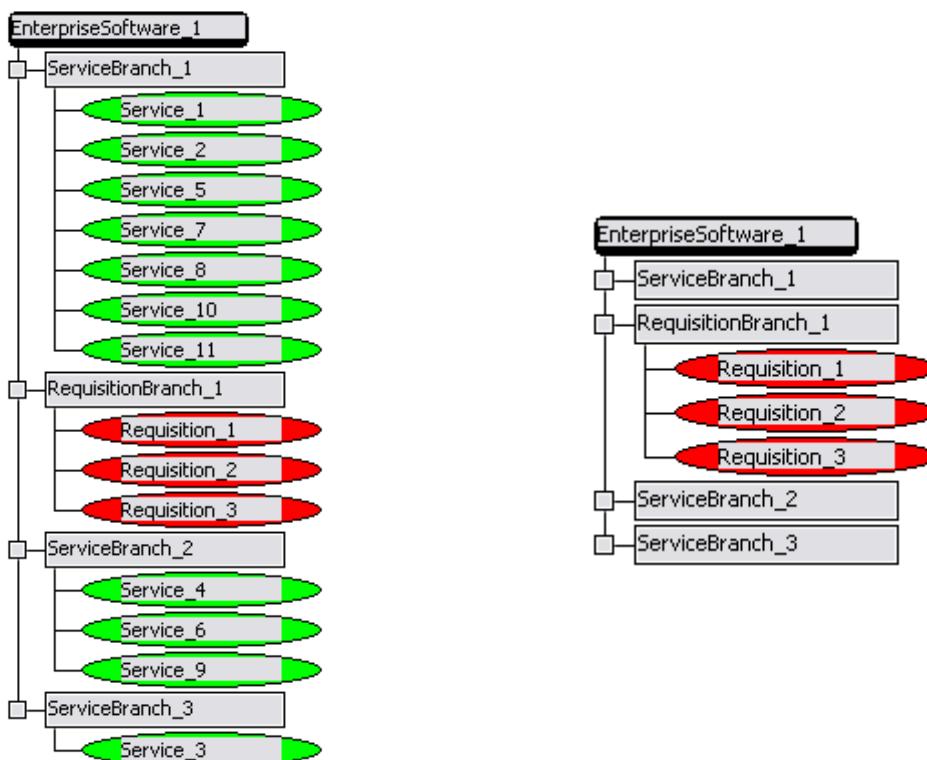


Figure 41: Tree Scalability.

Services and requisitions can be coupled in the business service contract diagram to represent service contract agreements. This is done by connecting a connector between a service and a requisition (**Figure 42**).



Figure 42: Service Contract Agreement.

The *ServiceFulfilmentContract* contains operations that must be fulfilled by the service. These operations may be derived from *sequential* business process models. To be able to display the order of the operations to be executed, in this service model, a flow chart based diagram is used. A vertical tree is used to display the order of artefacts. By using a tree it is possible to hide branches, so that one can focus on specific details or paths.

An integration diagram serves as a means to derive new services to the software view from requisitions made by *EnterpriseSoftwareParticipants*. By using an InstanceImporter that can be started from the context menu, requisitions from *EnterpriseSoftwareParticipants* can be imported into the diagram. New services are created in the same way as above. However, the new service has to be linked to a *SoftwareParticipant* on creation in order to assign it to the software domain.

Behaviour for diagrams in the software view is conceptually the same as for the business view.

Services and requisitions that have owners, but that aren't represented in the diagram, cannot be associated with a new owner. Hence, if a service or a requisition is dragged onto a participant, nothing will happen to in the diagram. The user will however be notified through a dialogue.

6.5. System Design

The system will comprise three sub systems:

- PROSERVE modelling tool
- A BPMN modelling tool
- A model transformation tool

The PROSERVE tool and the model transformation tool are integrated for the proof-of-concept. An open source BPMN modelling tool is incorporated in the solution, but any BPMN modelling tool can be used as long as it conforms to format requirements given by the transformation tool.

6.5.1. Components

The main components in the PROSERVE tool are outlined in **Figure 43**.

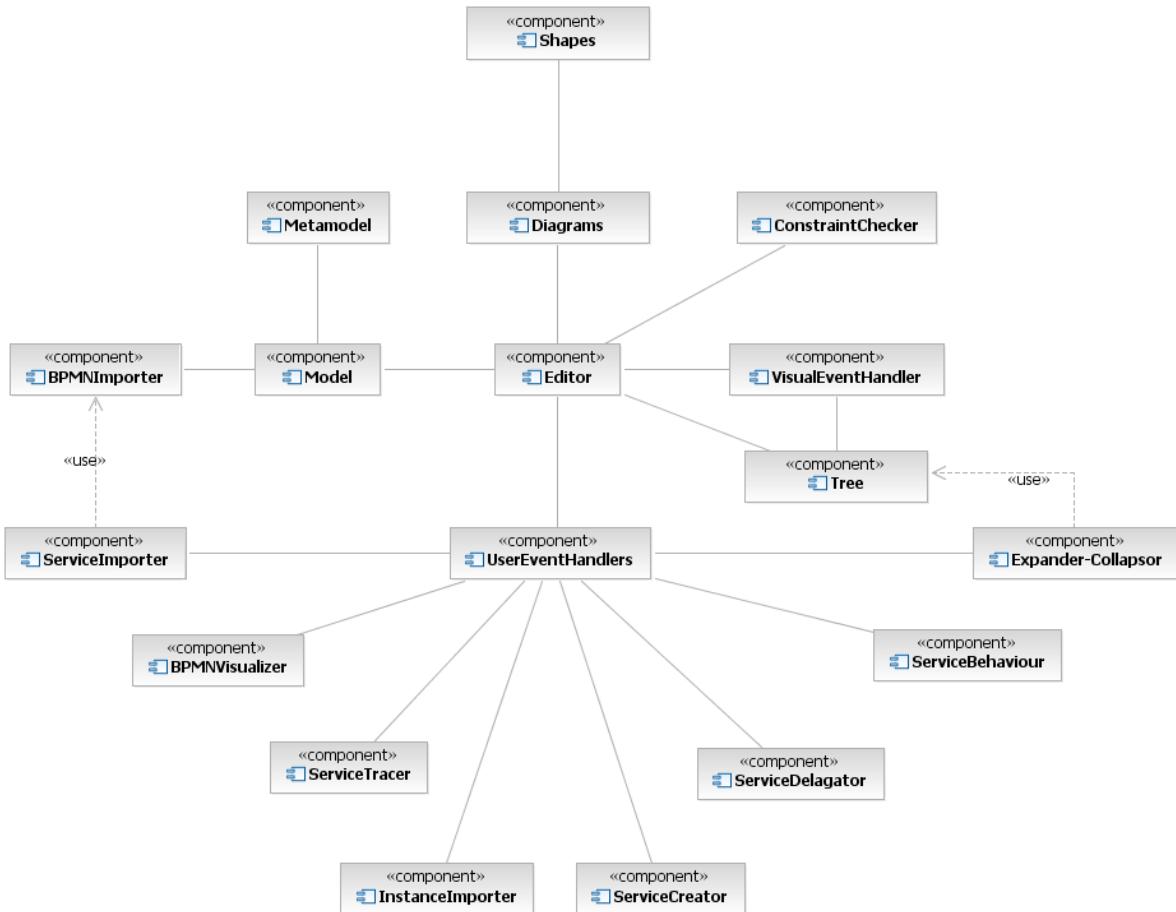


Figure 43: Main Components for PROSERVE Tool.

The core components are provided by the Marama platform:

- Metamodel
- Model
- Shapes
- Diagrams
- Visual event handlers
- User event handlers
- Editor

A metamodel defines the abstract syntax of the model. All model instances need to conform to this metamodel. Constraints that cannot be expressed in the metamodel are defined in the *ConstraintChecker*, e.g. service context constraints and diagram behaviour constraints.

The *diagram* component holds definitions of the different diagrams, i.e. allowed shapes and couplings to entities in the metamodel. Shapes that are tree nodes have special properties that are needed to construct and manipulate the trees. These properties are described section 6.6.2.

The *editor* component displays different diagrams and let the user manipulate shapes that are defined for them. When a shape is added, an instance of the mapped metamodel entity of the shape is persisted to the model. If a shape is included in a tree, a new association is persisted as well.

A *tree* component is invoked from a visual event handler. The component is responsible for building trees. It is dependent on constraint checking to check whether two nodes can be connected. **Figure 44** shows steps to build a tree.

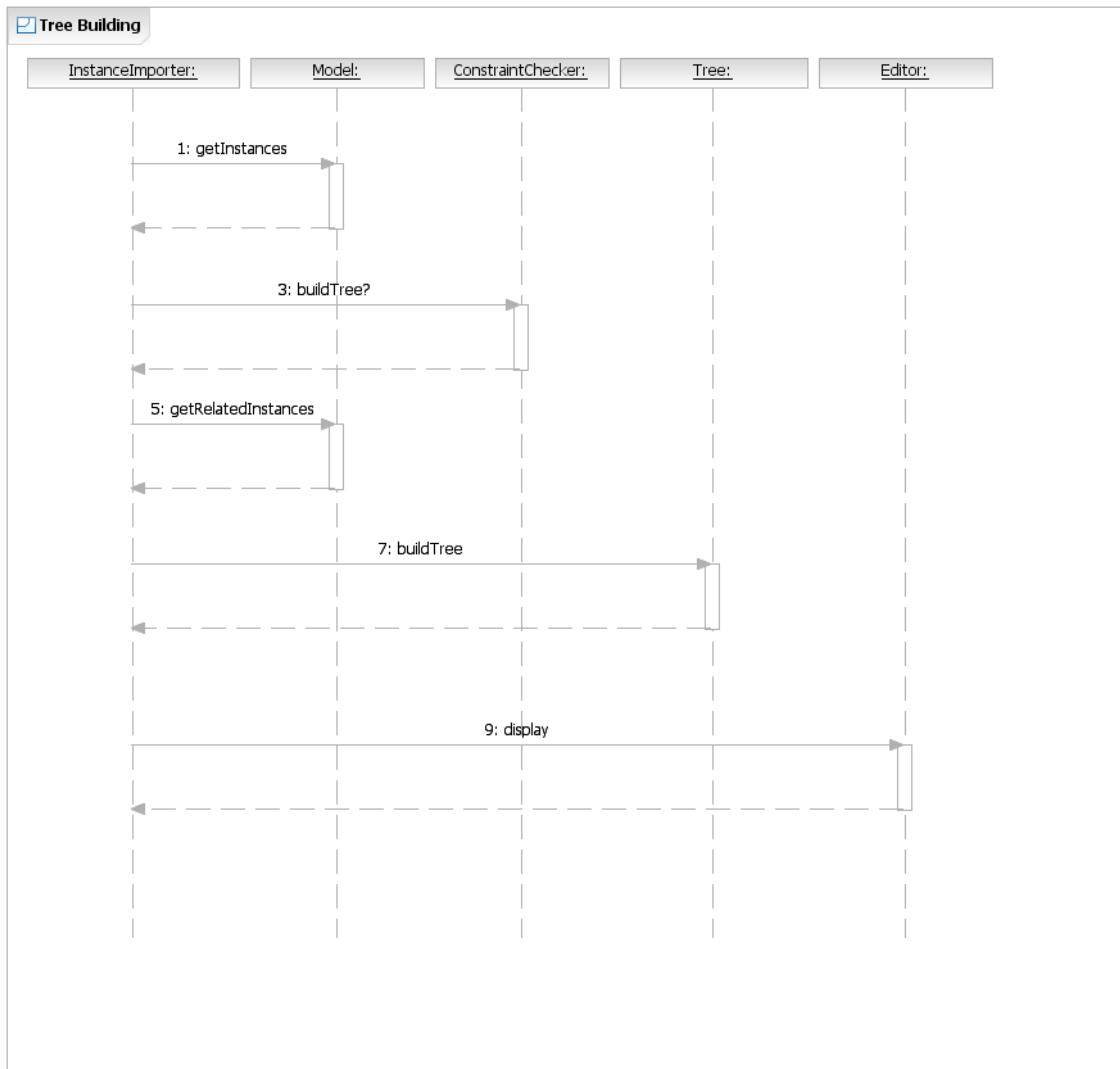


Figure 44: *Tree Building on Import.*

A *BPMNImporter* is responsible for deriving content to PROSERVE. It feeds a BPMN model to a model transformer and creates new artefacts for PROSERVE based on mapping rules described in section 5.2.

EventHandlers allow specification of diagram behaviour of two types. Any change in a diagram raises an event that is checked in a *visual event handler*. Defined conditions trigger specific behaviour. It is checked for tree creation for every movement or creation of a shape in the diagram. The second kind of event handler is *user event handlers* that are invoked from context menus in the editor.

6.5.2. User Event Handlers

This section describes the user event handlers that are implemented in the tool.

ServiceImporter

The ServiceImporter is implemented in order to offer the capability of identifying services from a business process model. A wizard is created so that the BPMN file can be selected. The request for service derivation is then made to the BPMNImporter which is responsible for the model transformation.

Instance Importer

In order to access instances in the model, an importer is implemented. The importer lists available instances that are allowed to be visualized in the diagram according to the service context scheme. By doing a differentiation between instances represented in the diagram and the ones existing in the underlying model, a list of import candidates can be generated (**Figure 45**). Shapes that are created on import are placed in the upper left corner of the diagram.

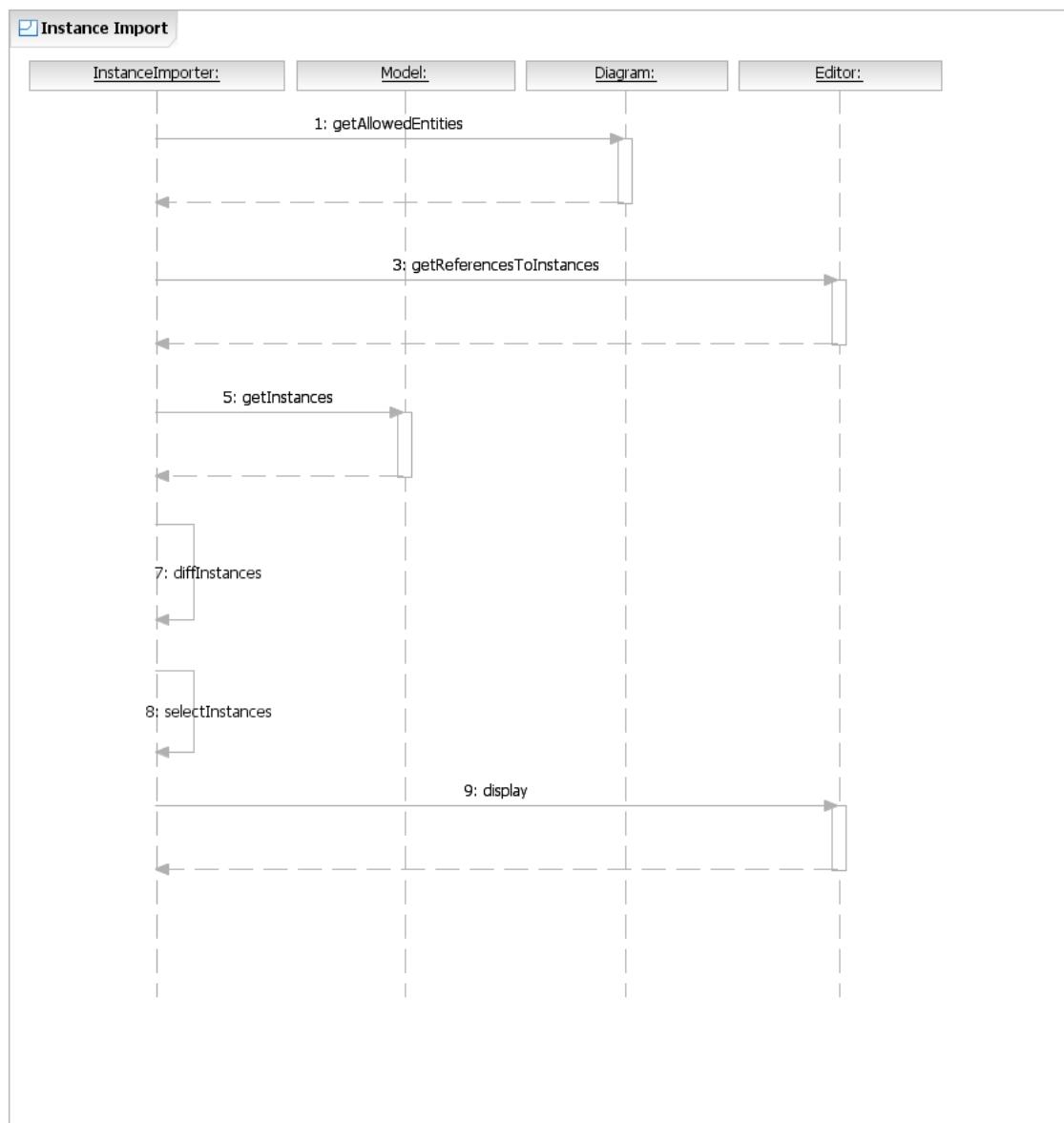


Figure 45: Importing Instances from Model.

Create Service Delegation

An important concept in PROSERVE metamodel is service delegation, which is a mechanism to decompose services and to delegate deliveries for them. By right-clicking on a service, “*Create Service Delegation – Enterprise*” can be selected to derive a new *EnterpriseServiceDelegation* and a new Requisition. In order to derive a new Service a *ServiceFulfilmentContract* must be added to the Requisition.

Create Service

In order for a service and a requisition to be allowed to connect, the requisition must propose a *ServiceContract*. By dragging a *ServiceFulfilmentContract* on top of a requisition, a service can be derived by selecting “*create service*” from the context menu. It will result in a new service to be displayed that has a connection to the requisition it was derived from. The contract is now made.

Expand-Collapse

To enable hiding of branches in service trees, “*expand-collapse*” can be selected from the context menu. If a node has a control shape it means that its branches can be collapsed. The same event handler is selected when a branch needs be expanded.

BPMN Visualizer

After populating PROSERVE with artefacts derived from BPMN models, it is possible to visualize these. Since Service has a Boolean attribute *BPMNderived*, the model can be parsed to display these services and related elements. BPMN Visualizer is more powerful than the instance importer since it builds complete service trees. By querying the model, related elements are found. Then the trees are built and distributed in the diagram. Trees are displayed in a collapsed state.

Service Tracer

In order to analyse impact of change, services can be traced to visualize dependencies. By selecting “*trace superior services*”, PROSERVE displays a vertical tree that represents chains of services leading up to its root service. On the other hand, selecting “*trace inferior services*” traces services that are derived from a given service.

Service Behaviour

Business process models describe activities sequentially. Service Behaviour can re-create order of activities in a service model context. Since ordering information is persisted to operation and requisition artefacts in the model transformation, it is possible to build these sequences again. The sequences visualized in a service behaviour diagram as vertical trees.

6.5.3. BPMN Modelling Tool

A BPMN modelling tool is needed to prove the concept of content derivation. An existing open source tool is used for this. Since an Ecore based transformation tool is selected, this tools needs to persist models in Ecore-format.

6.5.4. Model Transformer

The objective for the model transformer is to parse a BPMN model and derive PROSERVE artefacts based on its transformation rules. The tool selected for this task is described above.

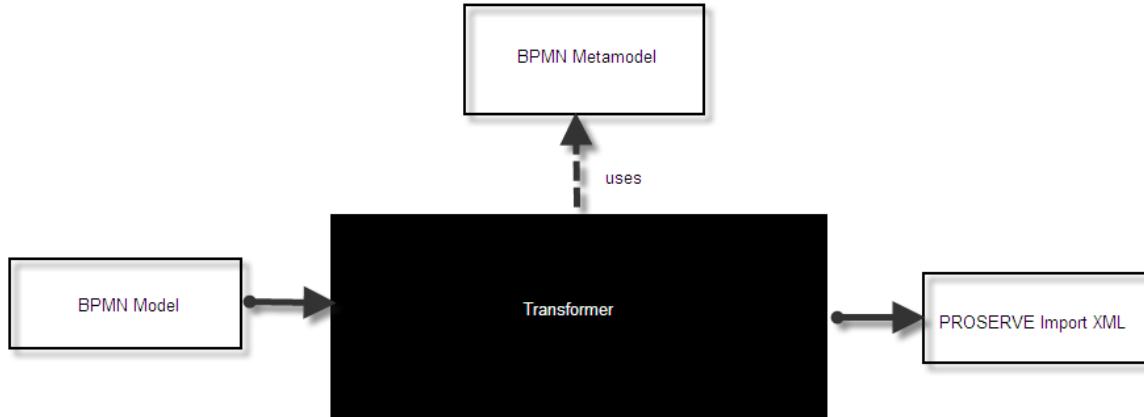


Figure 46: BPMN to Service Transformation.

The model transformer needs to perform the task of executing transformation rules on BPMN models. A BPMN metamodel is needed for rule execution and model validation. A temporary PROSERVE XML is generated. It is then possible for to generate the PROSERVE model by using the API of the Meta-tool (**Figure 46**).

6.6. Realization Overview

Marama comes with a visual tool for specifying a metamodel. The PROSERVE metamodel specification denoted in UML/MOF syntax in section [5.1.6](#) is used as input for the metamodel realization in Marama.

Shapes are defined in a GEF based editor in Marama. A shape-type consists of layout properties that define its visual notation. In addition, shape properties can be defined and displayed visually. These properties can be mapped to attributes in the model. They are also used to store meta-information for trees in diagrams.

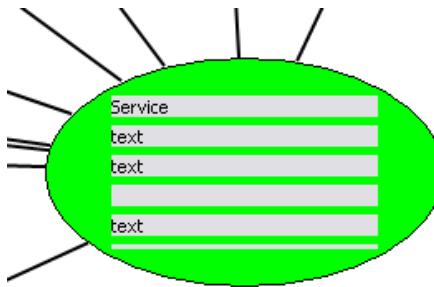


Figure 47: Shape Definitions in Marama.

In Marama, a *viewtype* or diagram is realized by linking shapes to entities. The *viewtype* file defines allowed shapes in the diagram and mappings to the underlying metamodel (**Figure 48**).



Figure 48: Linking Shapes to Entities.

Diagram types can use two kinds of event handlers. A visual user handler is triggered from a context menu, while a visual event handler is triggered by any movement of shapes in a diagram. **Table 8** gives an overview over event handlers that are implemented. A visual tree event handler was created with help from the Marama community at the University of Auckland.

Event-handler	Event-Handler type
Visual tree handler	Visual event handler
Collapse-expand branch	Visual user handler
Model instance importer	Visual user handler
Service Delegator	Visual user handler
Service Tracer	Visual user handler
Service Creator	Visual user handler

Table 8: Event Handlers.

Event handlers, BPMNImporter and ConstraintChecker are implemented in Java code.

6.6.1. Transformer Integration

The transformation tool is integrated with the BPMNImporter in PROSERVE tool. MOFScript comes with a Java API [67] which enables runtime specification of an input model file and an output file location. The BPMNImporter is wrapped in an Eclipse-Marama wizard for creating a model project. Since file paths need to be specified in the Marama format, a post-transformational XML DOM [68] manipulation is done to insert this.

6.6.2. Tree Algorithm

A tree is constructed from tree nodes. A simplification of the tree metamodel in **Figure 49** is made, so that all nodes are of the same type (**Figure 50**).

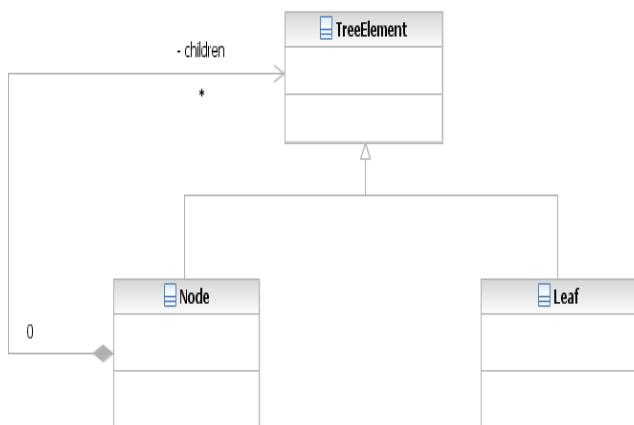


Figure 49: Tree Metamodel. Adapted from [69].

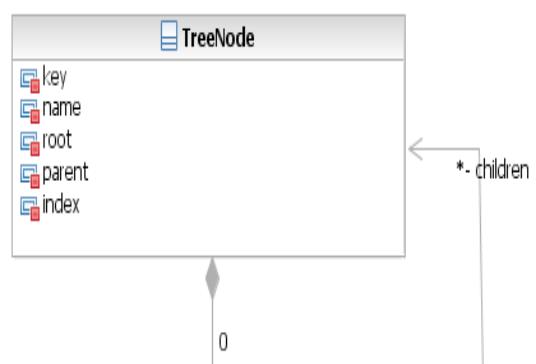


Figure 50: Simple Tree.

The intention was originally to use the tree only as a part of the concrete syntax. However, in the Marama version used a shape had to be coupled with an entity in the metamodel in order to be displayed in a diagram. Although this causes pollution of the metamodel, the only implication is some overhead information in the model. Currently, a new version of the tree implementation has been made in Marama that make the tree only a part of the notational syntax²⁷.

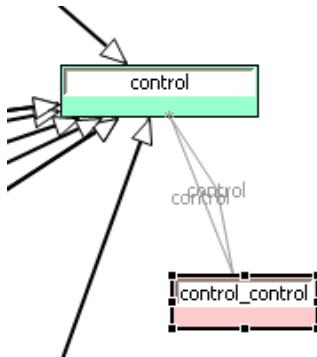


Figure 51: Inheritance from *Control* is Necessary to Make an Entity a Tree Node.

In the metamodel, a tree node is a *control* entity with an association to itself. An entity can be made a tree node by inheriting from *control*. The attributes in **Figure 50** are implemented in shapes as properties. Tree properties hold information about its position in the tree, its root and parent (**Table 9**). Since the meta-information is stored in shape properties, it is diagram-specific. This enables service trees to be built differently from diagram to diagram.

Property	Description
Name	Id for the tree-node shape
Depth	Is the level distance from the root node to the branch of the node
Index	Is the order for the node in a branch
Root	The Name of the root node
Parent	The Name of the parent node

Table 9: Tree-Node Properties.

Finally, the tree implements an algorithm for resizing of its node. By changing the size of the root node, all other nodes are set to the same size. This represents a zooming mechanism for trees.

6.6.3. Model Transformation

Transformation rules are implemented in MOFScript language [67] based on the transformation rules outlined in section 5.2. By querying the BPMN model, artefacts for PROSERVE can be derived. Recursive methods are created to derive the successor attribute needed for service behaviour.

²⁷ Honors work to be done by Pei-Shan Yap at the University of Auckland

```
336 bpmn.SequenceEdge::FindTarget(edge:bpmn.SequenceEdge){  
337     if(edge.target.activityType == 'Task'){  
338         '<successor>'  
339         '<name>' + edge.target.name + '</name>'  
340         '<ID>' + edge.target.iD + '</ID>'  
341         '</successor>'  
342     }  
343  
344     else if(edge.target.activityType == 'SubProcess'){  
345         '<successor>'  
346         '<name>' + edge.target.name + '</name>'  
347         '<ID>' + edge.target.iD + '</ID>'  
348         '</successor>'  
349     }  
350     else if(edge.target.activityType == 'GatewayDataBasedExclusive'){  
351         edge.target.outgoingEdges->forEach(outgoing:bpmn.SequenceEdge){  
352             outgoing.FindTarget(outgoing)  
353         }  
354     }  
355 }
```

Figure 52: Part of Rules for Deriving Behaviour.

6.7. Summary

This chapter outlined a technical solution for the prototype that was made to prove concepts in PROSERVE metamodel and the service tree mechanism.

Chapter 7. Design of Experiment, Tests and Analysis

The purpose of this experiment is to evaluate the requirements for the solution stated in Chapter 4 and validating the predictions defined in Chapter 3. First, the case study outlined in section 7.1 is extended in order to create a lager business process model. This model will be used to test the proposed solution. Section 7.2 and section 7.3 describe tests for the service context scheme and the PROSERVE tool, respectively. Implications of the solution on the case study are outlined in section 7.4. Finally, a cognitive dimensions analysis is conducted in section 7.5 in order to evaluate the service tree artefact.

The experiment involves processing the business process model in the PROSERVE tool. These will populate the business view. By deriving new services through the integration view, all views will be populated.

7.1. Appointment Process for Out-patients at a Hospital

The following business process is captured from Ullevål University Hospital²⁸. It depicts a simplified process from when a general practitioner (GP) requests an appointment for a patient until the appointment is completed. Such a scenario is complex, since it involves a huge number of tasks (the number depending on the level of abstraction used), interactions across the organizational border (e.g. patients and GPs), human concerns (e.g. social factors and HCI), tasks and rules implemented in technology, and contingency factors (e.g. worsening of a patient condition).

Since time does not allow a detailed description of this process, only a high-level process is depicted in a business process model. However, this is sufficient to demonstrate the concepts. After initial business service identification, new services can be created in the PROSERVE tool in order to extend the model to lower levels of abstraction, so that the concepts of context-views and context-interactions can be tested.

SOA Tools BPMN Modeler [60] is used to model the process. It is modelled without using any guidelines or methodology for business process modelling. The BPMN standard does not include a methodology.

An appointment process for out-patients consists of three main sub-processes as shown in **Figure 53**. The process is triggered by the event *ReferralReceived* which occurs when a GP's referral is registered at the hospital. In order for the appointment to be processed, the patient needs to show up (*PatientAttends* event).

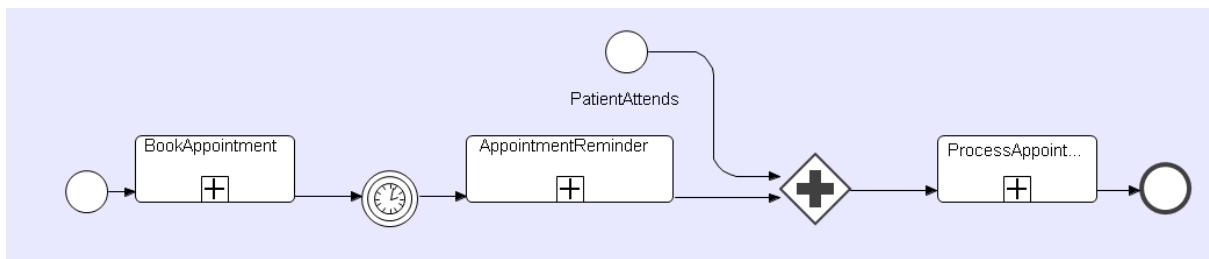


Figure 53: Appointment Process for Out-patients.

²⁸ Norway's largest hospital: <http://www.ullevaal.no/>

Book Appointment Sub-Process

BookAppointment is the first main sub-process. The purpose of this sub-process is to process the request from the GP and making a decision on whether the patient should be granted an appointment. **Figure 54** illustrates this process as chains of activities. *BookAppointment* encompasses the *AssessReferral* sub-process which captures a specialist's assessment of the referral. Details of this sub-process are black-boxed, i.e. the sub-process is empty.

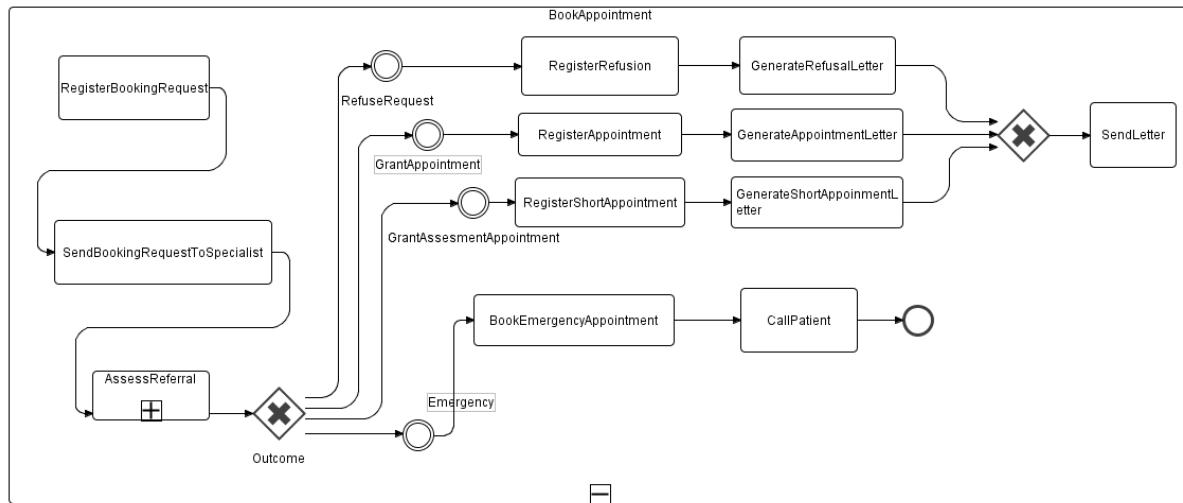


Figure 54: BookAppointment Sub-Process.

Appointment Reminder Sub-Process

AppointmentReminder is the next main sub-process. It describes the internal activities necessary to produce a reminder for a patient and exception handling. All the tasks that implement this process are wrapped in sub-processes (**Figure 55**).

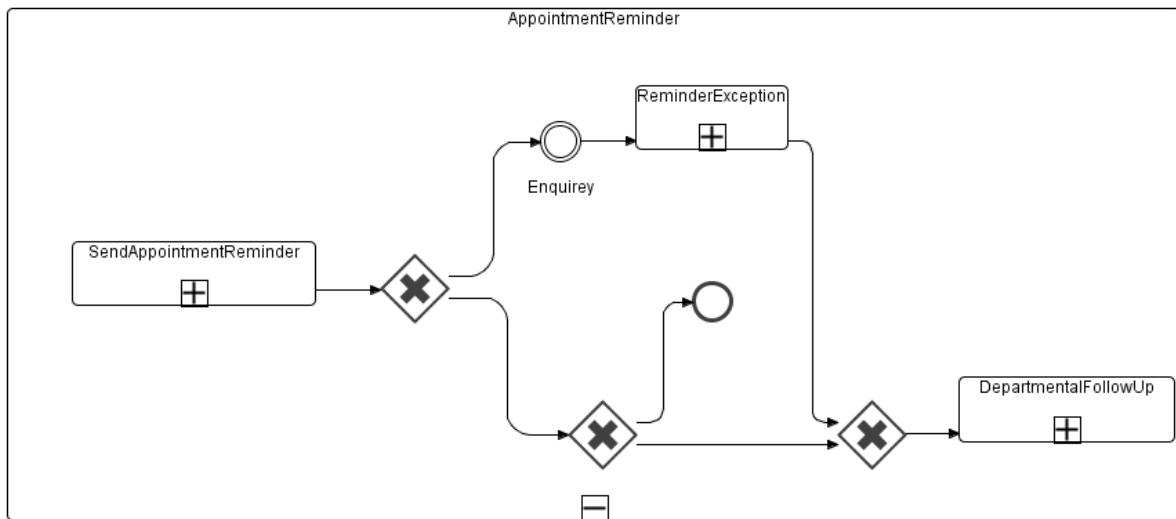


Figure 55: AppointmentReminder Sub-Process.

SendAppointmentReminder (**Figure 56**) is the first sub-process in *AppointmentReminder*. It depicts a sequence of activities resulting in a message being sent to the patient.

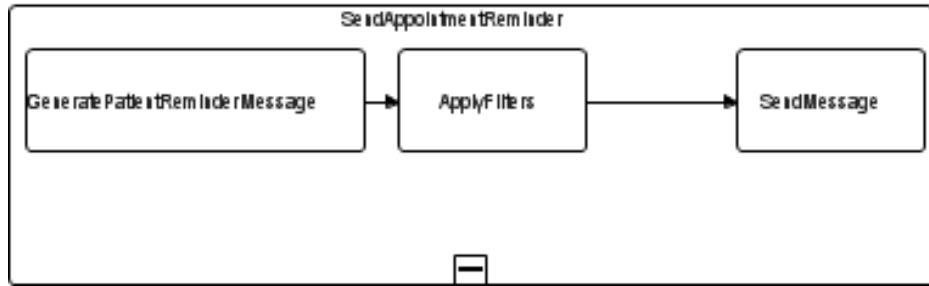


Figure 56: *SendAppointmentReminder*.

In a situation where a patient calls the hospital's switchboard to enquire about attendance to an appointment, the *ReminderException* sub-process is triggered²⁹ (see **Figure 57**). Enquiries should normally not be necessary, since the patient has received a letter about the appointment with all the information needed. However, the time from when this letter is sent and to this reminder is sent can be significantly long, e.g. a year. The patient may then have forgotten about it and the letter of appointment may have been lost. Moreover, the appointment may have been rescheduled multiple times, which can cause confusion. Finally, the wrong person may have received the reminder, e.g. when an incorrect mobile number has been registered.

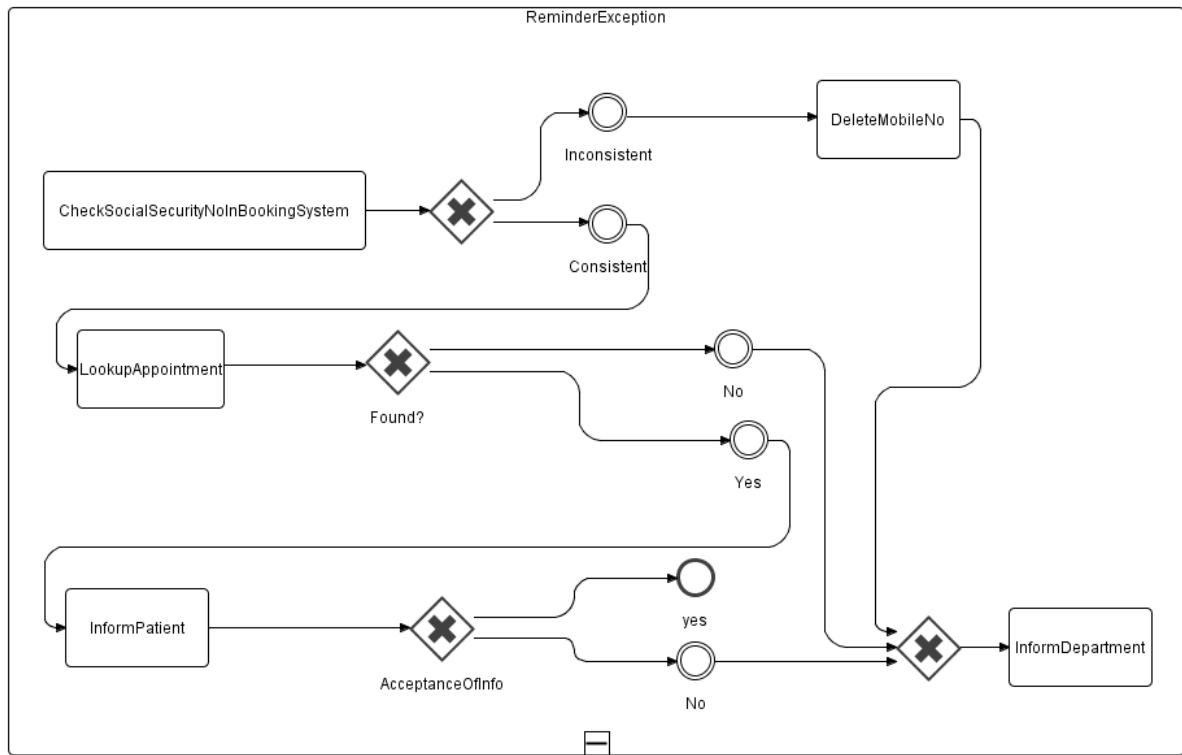


Figure 57: *ReminderException Sub-Process*.

In some cases, as depicted in the *ReminderException* sub-process, a clinical department needs to be informed about the exception that has occurred. This triggers the *DepartmentalFollowUp* sub-process (**Figure 58**).

²⁹ The rules of confidentiality constraining the type of information in messages were explained in section 3.1.

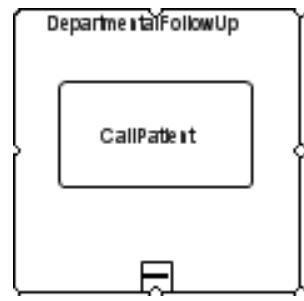


Figure 58: *DepartmentalFollowUp* Sub-Process.

Process Appointment Sub-Process

The last main sub-process in the appointment process is *ProcessAppointment*. It encapsulates sub-processes needed in order to carry-out the appointment (**Figure 59**).

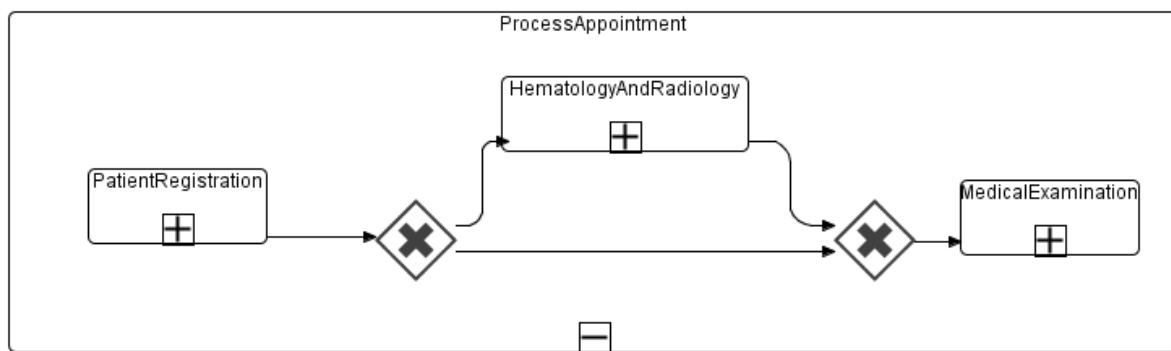


Figure 59: *ProcessAppointment* Sub-Process.

When a patient attends an appointment, the hospital registers him or her in as a patient. This includes registering necessary metadata for the consultation and follow-up (**Figure 60**).

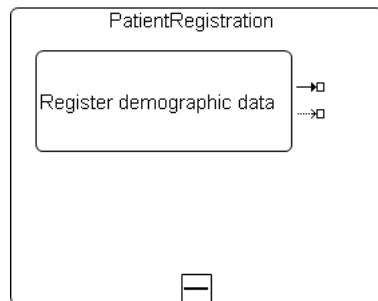


Figure 60: *PatientRegistration* Sub-Process.

Before a medical examination is processed, standard blood- and radiology-screening may have to be carried out (**Figure 61**).

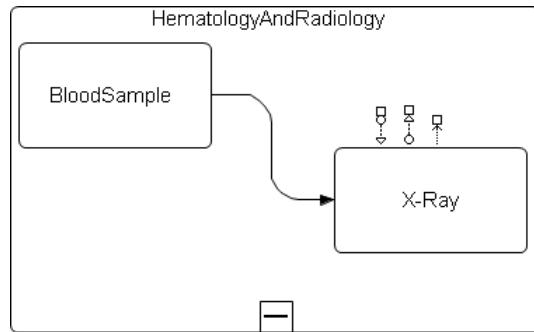


Figure 61: HematologyAndRadiology Sub-Process.

The last sub-process is the *MedicalExamination* (**Figure 62**), which represents execution of the actual appointment.

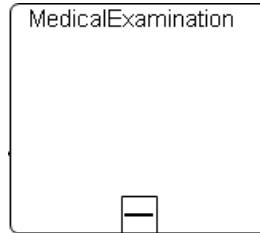


Figure 62: MedicalExamination Sub-Process.

The above BPMN-model is used as input for business service identification in the PROSERVE in order to test the proposed solution.

7.2. Tests for Service Context Scheme

In this section, it is explained how the tests for the service context scheme were carried out.

Scheme Requirement 1

The concept of assigning services to a domain by using participant ownerships was tested by processing the *Appointment Process* model in the PROSERVE tool to auto-generate service elements. Then, new services were derived manually in order to populate the software view. In order to test the concept of context-views, service and requisition instances from the underlying model were attempted to be imported into diagrams in the different views.

row	id	key	connectorRequ...	name	hasOwner	key	bpmnDerived	ParticipantType	successors	bpmnId
+ 1	Service.10	Service.0.87478697563...	null	AssessReferral	null	0.87478697563...	true	null	null	_fXrrIEAoEd2...
+ 2	Service.11	Service.0.44811491594...	null	BookAppointment	null	0.44811491594...	true	null	null	_7-VRkD2uEd2...
+ 3	Service.12	Service.0.12673629198...	null	PatientRegistra...	null	0.12673629198...	true	null	null	_eREsD3FEd2...
+ 4	Service.13	Service.0.82956985687...	null	HematologyAnd...	null	0.82956985687...	true	null	null	_Uu51bD3GEd2...
+ 5	Service.14	Service.0.89667783072...	null	MedicalExamina...	null	0.89667783072...	true	null	null	_OOU_wD3GED...
+ 6	Service.15	Service.0.53247254927...	null	ProcessAppoint...	null	0.53247254927...	true	null	null	_t54kD3FEd2...
+ 7	Service.16	Service.0.64509210488...	null	SendAppointme...	null	0.64509210488...	true	null	null	_62vusD2_Ed2...
+ 8	Service.17	Service.0.37476713280...	null	ReminderExcept...	null	0.37476713280...	true	null	null	_71H7ID3CEd2...
+ 9	Service.18	Service.0.81758780839...	null	DepartmentalF...	null	0.81758780839...	true	null	null	_ePL-wD3HEd2...
+ 10	Service.19	Service.0.88173035381...	null	AppointmentRe...	null	0.88173035381...	true	null	null	_A1vvoD2zEd2...
+ 11	Service.20	Service.0.86184701835...	null	Out-PatientAp...	null	0.86184701835...	true	null	null	null

Figure 63: Generated Services.

Since the only possible way an existing instance can be represented in another diagram is through the *InstanceImporter*, it was sufficient to test that this facility implements the rules

defined by the service context scheme, i.e. it was tested that diagrams did not break any of these rules. Diagrams that span over all views did not have to be tested.

Scheme Requirement 2

It was simply tested that services could be assigned to the different participant types.

Scheme Requirement 3

This requirement was not realized in the tool. The requirement will be evaluated below.

7.3. Tests for PROSERVE Tool

In this section, it is explained how the PROSERVE tool was tested, with regards to the requirements in section [4.2](#).

Tool Requirement 1

This requirement only needed an assessment of the tools that were used for the development of the solution.

Tool Requirement 2

This requirement only needed an assessment of how the tool was realized.

Tool Requirement 3

This requirement only needed an assessment of the solution.

Tool Requirement 4

In order to show that the order of the target artefacts was the same as the corresponding artefacts in the source model, a manual correlation analysis between the two models was conducted and then the sequences were compared.

Tool Requirement 5

In order to test that *ServiceContracts* have been generated correctly, it had to be tested that all artefacts were generated and that they corresponded to artefacts in the source model.

Since ID numbers are transferred from the BPMN sub-processes to related artefacts in the target model, comparing these ID numbers was sufficient to ensure complete transformation of these artefacts.

Operations were compared to task activities in the BPMN model. They do not contain the ID number from its source, but the names of the operations in the experiment were unique and could thus be used for comparison.

Tool Requirement 6

It was tested that all instances related to a *ServiceContract* could be visualized and that the service tree could be extended and collapsed.

Tool Requirement 7

In the experiment, services populated both the business and the software view. It was tested that paths from a specific service to its root service³⁰ and the path to its descendants could be visually represented in a diagram. The order of the connected services was compared to a manual log of service delegations that were done in the experiment.

³⁰ A service that has no connection to a requisition

Tool Requirement 8

Context-views were tested in Scheme Requirement 1.

Tool Requirement 9

This requirement only needed an assessment of the solution.

7.4. Application on the Case Study for Out-patients at a Hospital

The main goal of this thesis is to create new or improved artefacts that provide technical support for representing service descriptions in the context they belong in. Here, it is shown how the tool can auto-identify business service and provide different types of contexts based in the business process outlined in section 7.1.

PROSERVE enables both an automated population of a PROSERVE model and an automatic layout displayed in diagrams. The following diagram can be automatically generated³¹ from the business process model for out-patients:

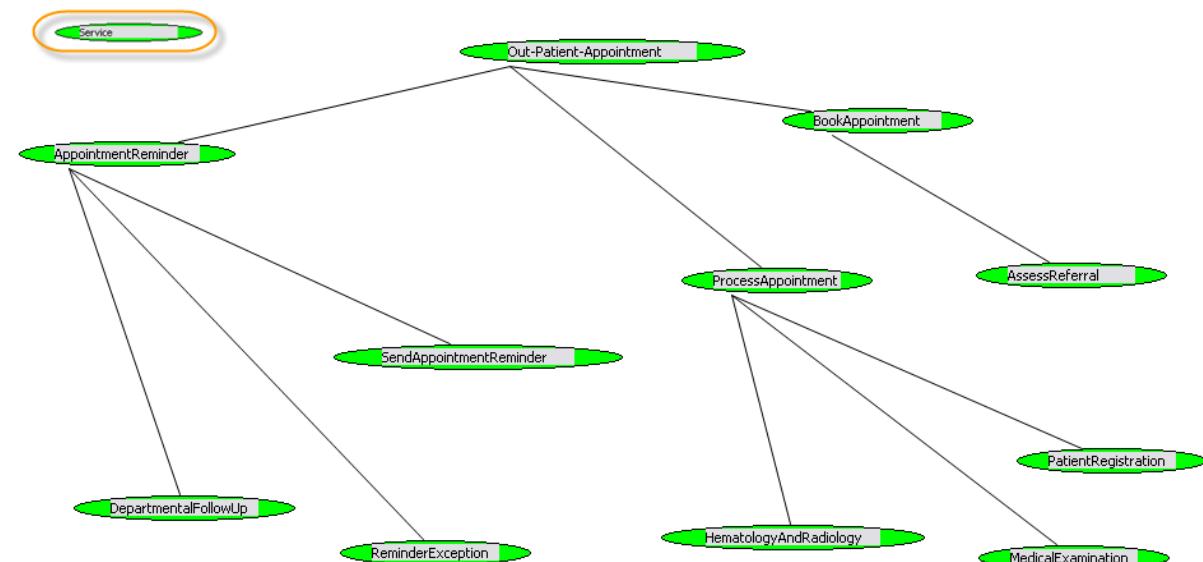


Figure 64: Automatic Visualization of derived Services in Business Service Contract Diagrams.

This view provides abstractions of the business processes dressed up as services. By expanding a service tree, the content of a service is shown, i.e. agreements on how delegated services are delivered. At the same time as the user can access details in the service abstraction, the universe they exist is also visualized (**Figure 65**). This is the concept of context-overview.

³¹ The elements have been moved to avoid crossing of connectors

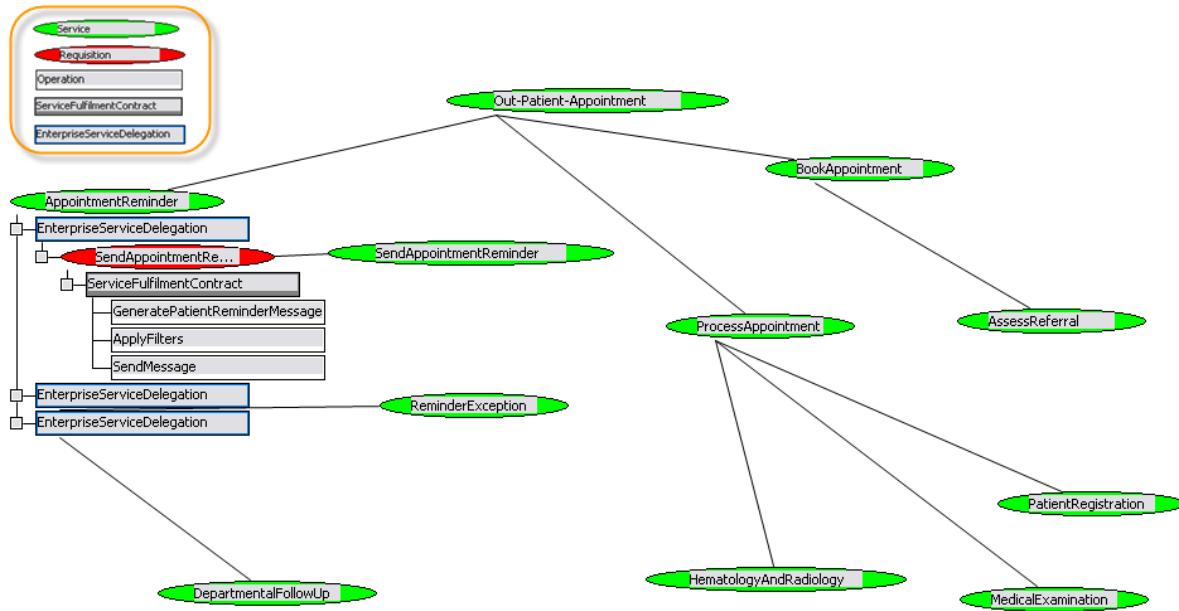


Figure 65: Selective Abstraction.

At this stage, strategic decisions in regards to which services will be delivered by IT can be made. Services that the hospital wants to outsource to IT are assigned to EnterpriseSoftware participants. On the other hand, services that are delivered by human effort are assigned to EnterpriseHuman participants. The following diagram presents possible decisions with regards to which services that can be outsourced to IT:

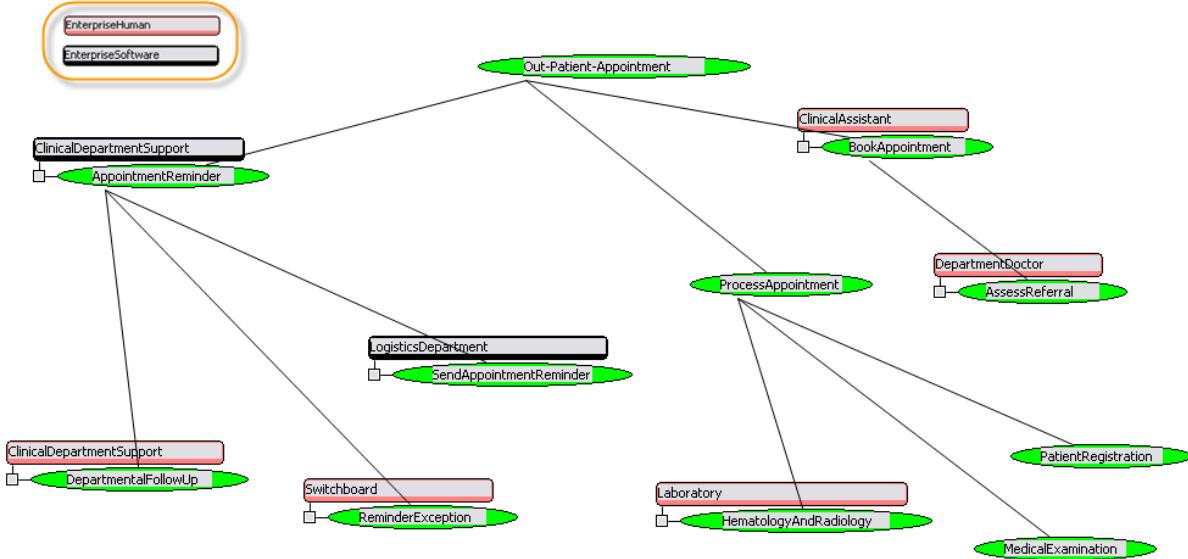


Figure 66: Adding Context through Ownerships.

In **Figure 66**, *AppointmentReminder* and *SendAppointmentReminder* are decided to be outsourced to IT.

Some services that are delivered by humans may rely on software services. A way to represent this is to make service delegations from human services. In this way, relationships

to supporting services can be described. For instance, in order for the clinical helper to complete appointment bookings, an electronic booking service may be needed (**Figure 67**). In PROSERVE, this is illustrated by making an *EnterpriseServiceDelegation* from the *BookAppointment* service. After defining a requisition and a *ServiceFulfilmentContract*, a new service can be derived. When this service is assigned to an *EnterpriseSoftwareParticipant*, it represents a service that is outsourced.

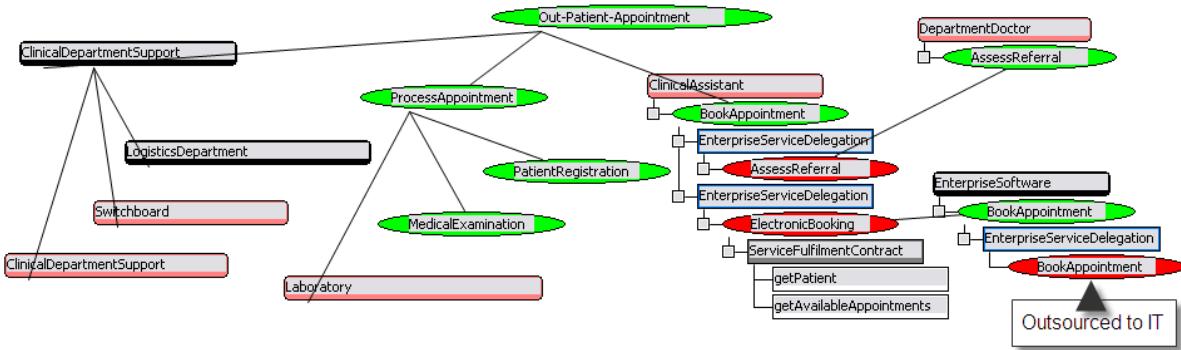


Figure 67: Outsourcing Service to IT.

Since IT may be delivered as services based on the SOA paradigm it is desirable to map these business definitions to representations of software services in a loosely coupled way. The solution provided in PROSERVE is to first describe business services and then delegating them via integration diagrams to a software view. Only requisitions that are owned by *EnterpriseSoftwareParticipants* are allowed to be imported in the integration diagram. By selecting “create service delegation” and “create service” from the context-menu, a new software service is created (**Figure 68**).

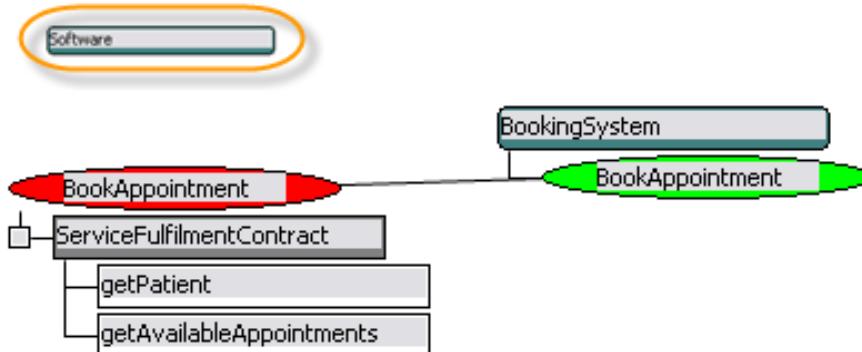


Figure 68: Delegating to Software Services in Integration View.

The software view gives an overview of software services that are necessary to provide services to business services in the business view. Software participants can represent software applications or components that provide the software services. In the context of software development, this thesis suggests that these service representations are mapped onto SOA models, e.g. UPMS for realization.

Traceability is a feature built in the PROSERVE tool to track dependencies. This can be useful for impact analysis of change.

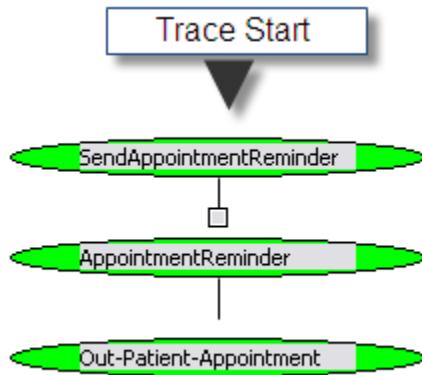


Figure 69: Tracing Superior Services.

Services can be traced in two directions. A superior trace can capture the chain of services from a given service to its root service (**Figure 69**). On the other hand, an inferior trace can capture all child chains of services from a given service (**Figure 70**).

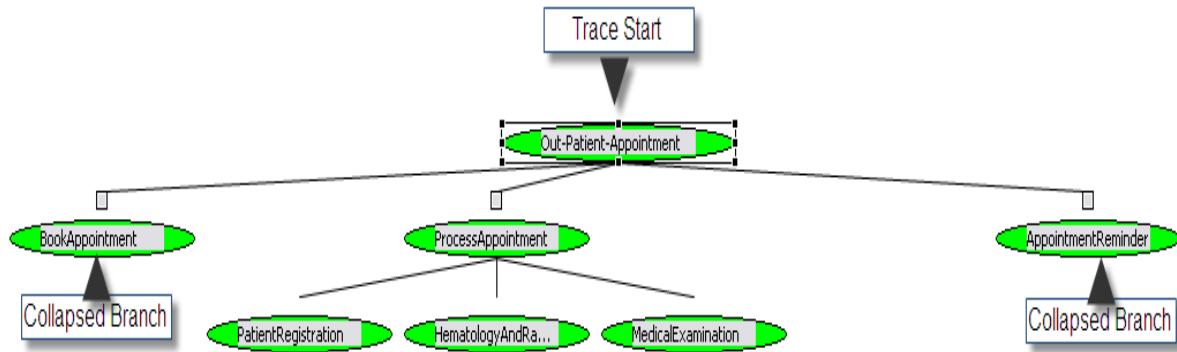


Figure 70: Tracing Inferior Services.

Finally, for services derived from business processes (BPMN), behaviour can be presented visually, i.e. the order of activities that produces the services. Requisitions in the diagram denote required services (**Figure 71**).

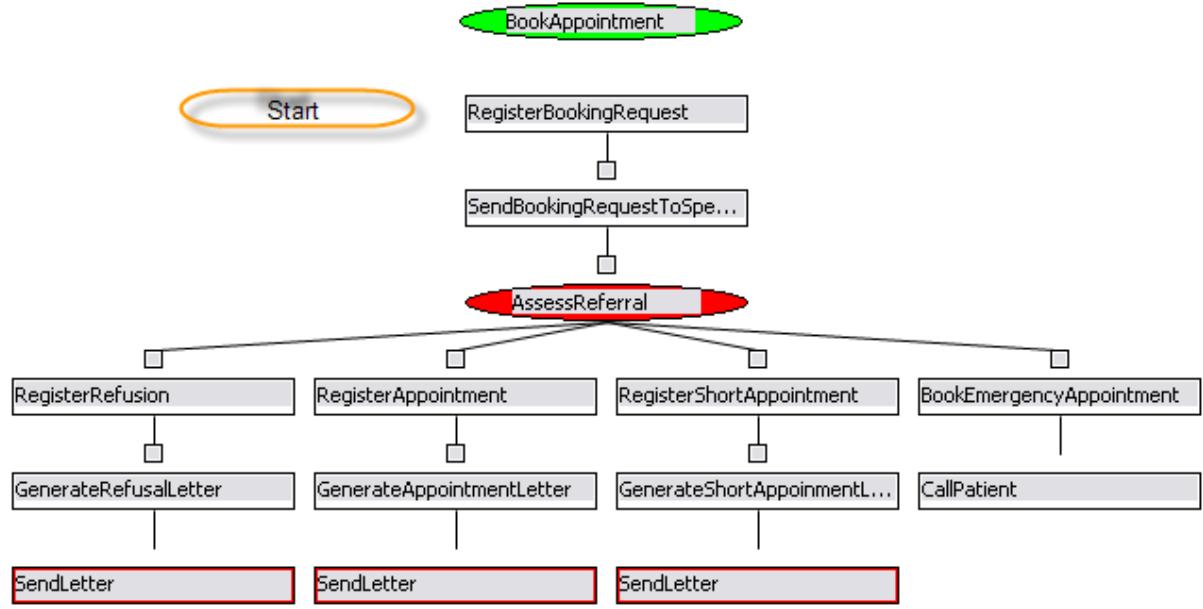


Figure 71: Service Behaviour of BookAppointment Service³².

7.5. Cognitive Dimensions Analysis of the Service Tree Structure

In order to evaluate an information structure's impact on users, an empirical study may be carried out. An alternative approach is to evaluate a structure of a diagram by using an analytical framework [70]. In this thesis a Cognitive Dimensions Framework (CDF) [71] is used to validate usability issues for the tree structure.

In order to assess usability of the diagrams in PROSERVE, a set of *cognitive dimensions* are considered. The purpose of the cognitive dimensions framework is to evaluate how well a notation support intended activities, given the environment and medium [72].

CDF considers types of activities that diagrams may support [73]. There are four groups of activities:

- Incrementation: Adding further information without altering the structure in any way.
- Transcription: Copying content from one structure to another structure.
- Modification: Changing an existing structure.
- Exploratory Design: Combining incrementation and modification.

Since PROSERVE tool only is a prototype intended to prove the above concepts, it does not support transcription and has only limited support for modification. Incrementation is therefore the only activity evaluated here.

Some dimensions in CDF are more focused on cognitive understanding. These dimensions are not used in the analysis in the same way as they are left out in the tutorial by Green and Blackwell [73].

³² An operation with a red border denotes multiple representations of the same instance

Viscosity

Viscosity is defined as resistance to change. The cost of adding an attribute in a model instance is low, since there are no dependencies to attributes in other model instances. Attributes that are represented in other diagrams than where it is added, are automatically updated.

Adding new elements to a service tree does not require other parts of the model to be changed. Propagating changes to other diagrams that already contain a service tree representation of the same service must be manually updated. This is a design choice that enables increased flexibility in what do display. Manual propagation does not itself increase viscosity, but it is potentially harmful (hidden dependency).

Hidden Dependencies

A hidden dependency occurs when two associated components are dependent on each other, but the dependency is not fully visible. It may lead to slow information finding because the dependency must be searched for.

The flexibility of manual propagation may lead to hidden dependencies. Typically, a service is presented in the diagram, but its owner is not presented. The user might then mistakenly try to assign ownership to an already owned service. To avoid this, an event handler disallows this and notifies the user.

The underlying model keeps tracks of dependencies. The model instance importer can be used to import new elements created in other diagrams to explicitly display the dependencies.

Premature Commitment and Enforced Lookahead

Premature commitment happens when a user is forced to make a decision before all information is available. *Enforced lookahead* describes a situation where a user has to look ahead in a cognitively expensive way.

Adding elements to the service tree does not require premature commitment, except for the action of creating a service for the software view. In order to clarify the context for the service, an owner (participant) must be added first and then the service must be created on top of it to create the ownership. In this way, a decision must be taken on which participant will own the service, before the service is created.

Abstraction

In CDF, an abstraction is described as a class of entities or a grouping of elements to be treated as one entity. Although, the service tree is not a model entity itself it represent a visual abstraction over a service and its related elements. Since the tool does not allow users to define new abstractions (abstraction-hating system), a further analysis of the abstraction dimension is not required.

Secondary Notation

Secondary notation is extra information carried by other means than the official syntax, e.g. annotations on diagrams. This is not provided in the proposed solution.

Visibility

Visibility is defined as the ability to view components easily.

The service tree provides a mechanism for information finding and selective visualization. According to Feldman, a tree is useful for depicting spatial relations among elements and groups of elements [74]. A tree consists of sub-trees which represent different groupings of elements which can be used for finding and viewing elements of interests. Moreover, a service tree has a specific ordering of its elements, e.g. if a service has child node, it is always a service delegation and if a service delegation has a child, it is always a requisition. The combination of ordering and partitioning in sub-trees enables a user to trace an element of interest from the root of the tree.

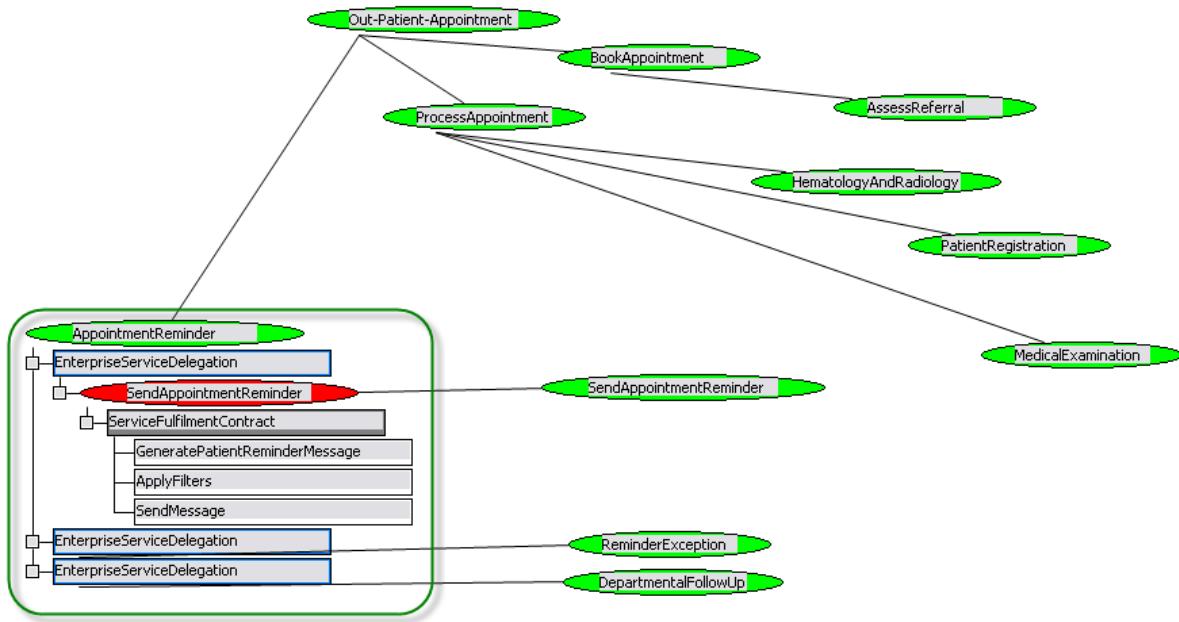


Figure 72: Scalable Information.

In addition, a user may collapse or extend any of the tree branches. Only information needed for navigation is unfolded as the user navigates to the target element.

Presenting a relationship between a service and a requisition which both are owned by the same participant, is creates cobweb problems. Although hidden dependencies are created, it can be better to remove this connector from the diagram, since it will improve the readability.

Juxtaposability

Juxtaposability is defined as the ability to place any two components side by side. This is a useful feature in order to do comparisons.

The diagrams support juxtaposability by allowing multiple service trees to be visualized adjacent to each other.

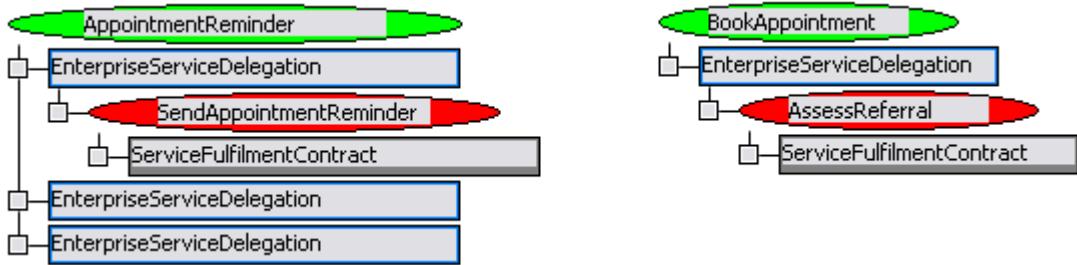


Figure 73: *Juxtaposibility*.

7.6. Summary

In this chapter, the solution was tested in an experiment based on the extension of the case study in section 3.1. This chapter also included a simple usability analysis.

Chapter 8. Evaluation and Discussion

The objectives of this chapter are to evaluate the proposed solution, criticize it and align it with related work. Section 8.1 and section 8.2 evaluate requirements for the service context scheme and the PROSERVE tool, respectively, based on tests in the previous chapter. In section 8.3, an evaluation of the service tree layout is conducted. Section 8.4 compares the proposed solution with state-of-the-art tools which were evaluated in section 4.3. The success criteria are then evaluated and the hypothesis is validated in section 8.5. Finally, the solution is aligned with related work and criticized.

8.1. Evaluation of Service Context Scheme Requirements

The evaluation will be based on the tests of the two first context scheme requirements in section 7.2. Scheme requirement 3 was not completely covered by the proof-of-concept and is therefore only evaluated assessed based on the theoretical solution.

Scheme Requirement 1

The scheme shall provide a solution for segmenting services and requisitions into a business service view and a software service view.

Fulfilment: Complete

The scheme was implemented in the tool which proofs the concept. Ownerships of services and requisitions through participant types were realized. Mandatory ownership for services in the software view requires enforced lookahead (section 7.5) which is regarded as a weakness. Still the requirement is met.

Scheme Requirement 2

The scheme shall provide a solution for differentiating between the natures of service delivery in the business view, i.e. whether they are provided by humans or by information- and telecommunication technology (ICT).

Fulfilment: Complete

The tool shows that services can be assigned to either an instance which represents a human entity or to an instance which represents a software entity.

Scheme Requirement 3

The schema shall provide a solution for how services are allowed to interact (context-interaction).

Fulfilment: Complete

The proposed schema provides a solution for how services are allowed to interact. Legal interactions are described in a matrix (Figure 22). It is, however, only given a proof-of-concept for context-interaction between views. Interaction rules within the business view were not realized.

8.2. Evaluation of Tool Requirements

Tool Requirement 1

The tool shall be implemented in open source technology.

Fulfilment: Complete

The tool was realized on the Eclipse platform which can be used for free under the Eclipse Public License³³. The Eclipse-based Marama meta-tool and MOFScript are both open source tools.

Tool Requirement 2

The tool shall be metamodel-based.

Fulfilment: Complete

A metamodel for the tool is specified in Marama meta-tools. The PROSERVE tool validates model instances on its metamodel.

Tool Requirement 3

The tool shall differentiate between abstract and concrete syntax.

Fulfilment: Partial

When the tree was realized in the Marama tool-set, it was required that all shapes in diagrams were mapped to entities in a metamodel. For this reason, notational artefacts for the service tree had to be represented in the metamodel, i.e. the control shape and tree connectors. However, later work on Marama has shown that visual artefacts can be implemented without a metamodel representation³⁴. It should therefore be possible to clean up the metamodel.

Tool Requirement 4

The tool shall support service behaviour.

Fulfilment: Complete

The experiment showed that service behaviour can be derived from business process models and visually presented in diagrams.

Tool Requirement 5

The tool shall support automated service identification from a business process model.

Fulfilment: Complete

PROSERVE tool provides a solution for deriving service representations from a business process model by implementing a set of mapping rules and processing them through model-to-text transformation. A service importer module reads the target XML and populates the service model by using the Marama API. The mechanism was validated in the test.

Tool Requirement 6

The tool shall support selective abstractions for service contracts.

³³ <http://www.eclipse.org/legal/epl-v10.html>

³⁴ Work by Jun Huh

Fulfilment: Complete

The tool provides a hierarchical tree that can present different elements related to a *ServiceContract*. While collapsing of the tree hides details, expansion of selected parts of the tree enables zooming of details without compromising the context-overview. It was tested that the tree was correctly generated.

Tool Requirement 7

The tool shall support traceability of services.

Fulfilment: Complete

PROSERVE provides a solution for tracing services bi-directionally. The mechanism was validated in the test.

Tool Requirement 8

The tool shall support separation of context-viewpoints based on a common model.

Fulfilment: Complete

PROSERVE provides both a business and a software view on the model. This was achieved by implementing rules based on a service context scheme. The solution was validated in the test.

Tool Requirement 9

The tool shall use metamodel-based model-transformation.

The tool uses MOF model-to-text transformations in order to translate business processes to service definitions.

8.3. Cognitive Dimensions Evaluation

The cognitive dimensions analysis conducted in section 7.5 is restricted to incrementation activities. It is therefore not a complete evaluation of the tool. In a complete version of the tool, an analysis should include these types of activities as well.

The results of the cognitive dimensions analysis are given in the table below, with respect to the incrementation activity.

Cognitive Dimension	Incrementation
Viscosity	Acceptable
Hidden dependencies	Harmful
Premature commitment	Acceptable, harmful in software view
Enforced Lookahead	Acceptable
Abstraction	Acceptable
Visibility	Acceptable, readability improvements
Juxtaposibility	Acceptable

Table 10: Cognitive Dimensions Evaluation.

The main weakness of the diagrams in the tool is lack of propagation of changes between the diagrams. This creates hidden dependencies which may lead the user to attempt illegal

actions, e.g. assigning ownership to an already owned service. Even though an event handler constrains the action from actual happening, a user can make wrong assumptions when making decisions on the basis of what is visually presented.

The main strength of the tool is the visibility that the service tree provides. As described above, it provides a structure for finding and viewing associated elements based on segmentation and ordering of elements.

8.4. Comparison with State-Of-The-Art Solutions

In order to assess the contribution of PROSERVE, it is compared with the results of the evaluation in section 4.3. A discussion of alignment with these tools is carried out in section 8.6.

0: Requirement is not fulfilled

1: Requirement is partially fulfilled

2: Requirement is fulfilled

Requirement	SOMA	COMET	EML	PROSERVE
TR1 Open Source Technology	1	2	2	2
TR2 Metamodel-based	2	2	2	2
TR3 Abstract vs. Concrete Syntax	2	2	1	1
TR4 Service Behaviour	2	1	2	2
TR5 Automated Process-driven Service Identification	0	0	0	2
TR6 Selective Abstractions	0	0	2	2
TR7 Service Traceability	1	0	0	2
TR8 Separation of Context-Viewpoints	0	0	0	2
TR9 Model-transformation	2	2	0	2
Sum	10	9	9	17

Table 11: Tool Comparison.

The comparison suggests that PROSERVE makes its primary contributions on automated-process-driven service identification, service traceability and separations of context-views.

8.5. Evaluation of Success Criteria and Hypothesis

The purpose of the success criteria is to validate the hypothesis defined in section 3.4. This is done by evaluating the predictions made in section 3.5.

8.5.1. Success Criteria

Success Criterion 1 (P-1): *The tool can provide a service representation of a business process model in a way that retains the order of activities.*

Fulfilled

A solution for mapping of core artefacts from BPMN to PROSERVE was described in section 5.2. This was implemented in the PROSERVE tool which enables PROSERVE model population from BPMN models and automatic visual presentation of these.

The order of activities in the BPMN model was successfully derived to the attribute *successors* in *Operations* and *Requisitions*. This was validated in the test for tool requirement 4.

Success Criterion 2 (P-2): *The tool can visually differentiate between services delivered by humans and services delivered by software in a business view.*

Fulfilled

An *EnterpriseHumanParticipant* represent human participant that deliver services, while *EnterpriseSoftwareParticipant* represent a software entity that delivers services. They are different entities in the metamodel and have different notation. By assigning services to these types of participants, the type of service delivery is determined.

Success Criterion 3 (P-3): *The tool can visually provide a mechanism for delegating services from a business view to a software view.*

Fulfilled

A generic service delegation mechanism was made for requesting sub-deliveries for a given service in the PROSERVE metamodel. When there is a need to delegate services to the software view, an integration view enables coupling between the views. The mechanism was validated in the test for scheme requirement 1.

Success Criterion 4 (P-4): *The tool can provide a software view where only services that represent software services are visualized.*

Fulfilled

This was successfully realized in the same way as for P-1, only by using *SoftwareParticipants* instead of enterprise components to extract the correct services from the underlying model. The concept was validated in the test for scheme requirement 1.

Success Criterion 5 (P-5): *Services in the tool can be traced bi-directionally across the domains to show dependencies.*

Fulfilled

A designated diagram for service tracing was made. In order to trace services, associations from a service are queried to see if it is connected with other services. While querying its associations to service delegations enables services to be traced down the stack, querying its associations to requisitions enables services to be traced in the opposite direction. Traceability was validated in the test for tool requirement 7.

Success Criterion 6 (P-6): *The tool can be used to visually present a service contract between a consumer and a provider of a service in a business view.*

Fulfilled

The business service contract diagram is able to present elements that make up a service contract. It is presented as two service trees where there exists an association between a requisition in one tree to a in the other.

Success Criterion 7 (P-7): *Service model instances can be presented as selective abstractions through a service tree in diagrams.*

Fulfilled

The notion of selective abstraction was realized as service trees that can be collapsed or expanded. These trees provide a means to focus on particular details at the same time as abstracting away other details, frees up more space to show surrounding artefacts.

8.5.2. Hypothesis

The success criteria acted as predicates to test the hypothesis. Since all of these were fulfilled, this thesis' hypothesis is validated, i.e. the following hypothesis is strengthened:

A business service modelling tool based on a service context scheme and a process-driven service identification will provide a means for presenting services in the context they belong to, so that SOA can be better coupled to business models through the use of the service metaphor in both domains.

A cognitive dimensions evaluation concluded that a service tree enhances the visibility in diagrams. By providing a mechanism for context-overviews, context support is strengthened.

This thesis can only argue that PROSERVE provides a solution for improved coupling between business models and SOA models. In particular, PROSERVE should be integrated with a SOA model in order to give a better evaluation of the hypothesis. This thesis main objective was to create a technical foundation for decision-making on outsourcing of services. Both the service context scheme and the process-driven business service identification can be potential technical means for creating decision-support. A future solution for this support needs to focus on the user perspective.

8.6. Discussion and Critique

The purpose of this section is to discuss PROSERVE's alignment to relevant artefacts and criticize both the PROSERVE solution and state-of-the-art technology.

8.6.1. PROSERVE Alignment with Existing Service Identification Approaches

PROSERVE uses the services metaphor as an artefact for business modelling within business architecture. It is used primarily in the meaning business-as-a-services (BaaS) [75], i.e. a service independent of technology. In traditional SOA, services are only described in context of service being software which can be referred to as software-as-a-service (SaaS) [75]. In SaaS, service is typically correlated to processes or use-cases in the business domain. Since SOA is limited to SaaS, an enterprise that primarily delivers intangible human services rather tangible goods, e.g. a hospital, SOA will not support service description of the core business. It is evident that the hospital should be able to describe its services for example to its patients, collaborators and internally in the organization at a business level. There are several arguments for disseminating the use of the service metaphor in the business domain and combining BaaS with SaaS in SOA:

1. Service identification can be made at a business level. The service term can be used to describe activities that facilitate exchange of values in a business and the organization's responsibilities to deliver these.
2. Business processes describes how a service can be delivered. It does not describe the delivery itself. It can in this respect be argued that a service is a higher abstraction than a process.
3. Human services need to be aligned with technical services. As an example, a hospital mediates its core services through human activities. Since IT services rarely represent core services themselves, it should be explain how they support them.
4. Service encapsulation of business can provide better grounds for identification of business that can be outsourced to IT. An identified service can potentially be a candidate for outsourcing or it can be de-composed into services that are suitable for this.

None of the state-of-the-art frameworks evaluated in this thesis represent services in a dedicated business context. PROSERVE can make a contribution to fill this hole.

In order to identify services in the business domain, a solution for automated process-driven identification of business services was created in PROSERVE. SOMA also supports service capturing from business processes. The main difference from PROSERVE is that the identification is not automated nor does it map to a BaaS representations, but directly to software-as-a-service (SaaS). Automation increases the speed of service identification. However, it is believed that in order to execute useful transformations, a tailored methodology on business process modelling is needed to describe activities with a granularity that is useful for capturing services. Another improvement in PROSERVE is that it retains the ordering of activities.

A goal-driven approach may co-exist with the process-driven approach in PROSERVE. In SOMA, business goals are refined into a level that is detailed enough for IT design and implementation. To adapt the approach to PROSERVE, business goals need not to be refined in such a detailed manor, since they are used for service identification at a higher abstraction layer. Some redundancy may be expected though, since goals can be linked to business processes.

Use-Case-driven service identification is supported by both SOMA and COMET. In SOMA, a use case is a precursor to a business process. Use-Case-driven identification can complement the PROSERVE's process-driven approach.

EML does not have a methodology for identifying services. It is in principle only an alternative notation for business process models, without definition of its own metamodel. The tool framework can also represent a business process in BPEL in BPMN notation. In this regard, it comprises a simple approach for service identification for software services. It is currently work in progress and needs more development and maturity in order to align it properly with other solutions.

PROSERVE may complement SOMA and COMET in order to support business service modelling. By incorporating the PROSERVE approach for service identification and context support, service networking is elevated to the business level. Such a supplement may help business people to utilize the metaphor as a contract and delivery mechanism both horizontally for the organization and vertically for service delivery from IT.

8.6.2. PROSERVE Alignment with Enterprise Architecture and Service-oriented Architecture

PROSERVE does only provide a simple solution for service and business process alignment in enterprise architecture (EA). Further work is needed both to improve the current alignment and to integrate other disciplines of enterprise architecture with business services. The service metaphor may glue together these dimensions.

Since a service is a mechanism for assigning responsibilities through service ownerships, it may be reasonable to map it to the organizational dimension of an EA. In this way, responsibilities can be anchored in the organization. It is therefore suggested that PROSERVE maps organizational units to participants in the model. Another approach may be to map participants to functional areas in an organization as done in SOMA.

The superstructure of an EA is descriptions of goals, strategies and tactics. This may be referred to as a motivational dimension. Motivation can be captured in a motivational model such as OMG's Business Motivation Model (BMM) [76]. UPMS proposes mappings from BMM to UPMS Service Contracts [77], but here it is suggested that the mappings are elevated to a higher abstraction level. PROSERVE metamodel does not have suitable artefacts for representing motivational concerns and needs to be extended in order to support them. In BMM, strategies and tactics are realized through a *course of action*. A course of action may again be realized by a business process. In principle, a *course of action* can also be mapped to a business service rather than to a business process.

Information objects in a business process model can be linked to information models in order to ensure consistency of information in an enterprise. This is for example supported in Telelogic's System Architect³⁵. The same may be accomplished for information objects in a service model. Since PROSERVE lacks information objects in its metamodel, it should be extended to support descriptions of information exchanged in service interactions. Traceability links can enable a model-driven update of information objects between service models and information models and vice versa.

In PROSERVE, applications can be represented as software participants in a software view. In fact, the view may be called application view if the primary task is to create mappings to logical representations of applications. However, the vision is to create deeper mappings to real software design specifications in future work.

Existing applications services may have to be identified. PROSERVE outlines a top-down approach for creating software view services. However, ideally, it should also be possible to identify software services from existing IT portfolios. A bottom-up approach is supported in SOMA through an analysis of existing assets. Coupling of business definitions with application services is a meet-in-the-middle approach. PROSERVE can improve this approach by doing a services to service mapping rather than a mapping from a service to multiple business artefacts such as goals, processes, organization and information. It is therefore suggested that PROSERVE is extended in order to support a “*meet-in-the-middle*” approach for existing application services.

A future challenge is to couple the approach in this thesis to SOA models, e.g. UPMS. The goal is to enable flow of business requirements into actual software development. Since

³⁵ System Architect: <http://www.telelogic.com/Products/systemarchitect/>

PROSERVE is built on some of the core concepts in UPMS, it is suggested that meaningful transformation rules are created in a way that can populate UPMS models. In this way, an end-to-end traceability from high-level business concerns to software architecture specifications and vice versa can be supported. However, a transformation from PROSERVE to UPMS might not be trivial since business definitions have to be reconciled with software design.

8.6.3. PROSERVE Alignment with Software Development Process

In traditional software engineering, requirements for a software solution are captured in a software development process. A software requirement engineering process is complementary to requirements derived in EA. A service definition should embrace or trace all requirements. PROSERVE may help keeping these concerns separated through context-views.

One approach can be to enrich services in PROSERVE's software view with software requirements. Business services in the business view will still be clean of these software specific concerns. Alternatively, enrichment with software concerns can be done in a SOA model. There is an obvious need to reconcile the EA-driven service approach with traditional software requirement engineering to correlate software requirements to business requirements. Some of these requirements may have cross-cutting concerns, e.g. non-functional requirements such as performance and security. This discussion is, however, too comprehensive to carry out here

In the Unified Software Development Process [78], use-cases capture system functionality and requirements in UML. Above it was described that use-cases can be used to identify services. However, it may also be possible to identify use-cases from services. Derived use-case may then be a part of the service description or linked to it.

8.6.4. PROSERVE weakness

PROSERVE proposes a solution for the needs identified in the problem analysis. It is not intended to be a tool for commercial use. Neither the metamodel nor the tool is full-scaled. Moreover, due to shortage of time some parts of the metamodel were not realized in the tool.

In regards to the evaluation of requirements, the main weakness is partial lack of separation between the notation and the metamodel. At the time of implementation, Marama Meta-tools required that every shape was mapped to an entity in the metamodel. But as mentioned, this requirement has been elicited so that the tree could now be implemented without pollution of the metamodel.

PROSERVE does not provide a solution for how pools and lanes in BPMN can be re-used in a service model. It is believed that other concerns in enterprise architectures have to be considered too in order to find a useful solution for how to use this information in another context. Since pools and lanes may represent organizational responsibilities, it makes sense to also consider models for organizational structure and how these three modelling approaches may be combined.

For the notation, the tool is not able to present service delivery and consumption from the same participant in a visually pleasing way, i.e. connectors are partially hidden by other artefacts. Moreover, the automatic layout of services identified from business process has

connectors crossing each other. Finally, a colour dependent notation may be a problem if the models are presented in other media. Further work on notation and layout is needed.

PROSERVE is a solution within the scope of enterprise architectures. It does not consider requirements from the software engineering side. So the actual coupling with software service descriptions in SOA models is not supported. The value of the integration view is therefore questionable. On the other hand, it might be useful to take into account software service concerns at this level in order to prepare for concerns in the SOA model. Software view services can then act as buffers between business definitions and SOA definitions. Further research is needed.

For PROSERVE to be a useful tool for service identification, a methodology is needed for business process modelling. Since services are automatically identified, processes need to be carefully designed. The value of automatic process-driven identification of services may be questioned because it indirectly requires that the modeller makes consideration about how services are identified. It can then be argued that the actual identification of the service is done in the business process model in a manual way. However, the mapping rules may be used as analytical guidelines for identification of services.

The analysis of state-of-the-art analysis is primarily focused on tools that have evolved from a software engineering perspective. A future evaluation should also include comparisons with enterprise architecture frameworks, for example Archimate³⁶.

8.6.5. Gained Knowledge

Service and Process Harmonization

The work in this thesis raises the questions:

*How are services and business processes aligned?
When do we need to describe processes?*

The common way of identifying *software* services in SOA is through processes. This is for instance a concept recognized by National ICT for Healthcare (NIKT) [79]. However, from the discussion above it may be claimed that the service identification should be lifted to a business level in order to incorporate several disciplines of enterprise architecture, e.g. motivation, business planning and coproduction of values.

Mapping the order of activities in an organization like a hospital to a process model can be extremely complex. While business process modelling has diffused well into production of goods where the order of activities is fairly standardized, it may be harder in the world of hospitals. Hospitals are dominated by human services where the order of activities often is quite opportunistic. It may for example depend on individual clinical assessment of a wide range of metrics that potentially trigger a huge amount of tasks that again may be executed in a seemingly random order. This makes it probably feasible only to model processes at a very high level of abstraction or in quite limited segments of a process for a certain purpose. It is also assumed that processes change more often than service deliveries, giving business process models shorter time to live. The last assumption needs validation, however. It is suggested that business process modelling is done on a need to do basis.

³⁶ <http://www.archimate.org/>

A service can define wants and needs from entities which an enterprise do business with. Since it can hold information about which values that are exchanged between entities, it can depict the motivation in a business. Processes, on the other hand, are not intended to describe their motivation. Hence, when describing the business of an enterprise from a motivational viewpoint in a top-down way, it is suggested that one starts with defining services across the organizational border that drives the business.

PROSERVE outlines not only a process-driven approach, but also a service-driven approach through a divide-and-conquer concept. After identifying top level business services, these services can be de-composed using the service delegation mechanism in PROSERVE. In this way, service deliveries are recursively realized by other services. As long as ordering of activities in the business is not needed, process descriptions can be excluded.

When it is necessary to represent ordering, a process description can be embedded in a contract. This was partially done in the *ServiceFulfilmentContract* in PROSERVE. In this way, the process is represented in its business context, i.e. in the network of services. The approach enables hospitals to describe its business as services and detail out processes only when it is necessary. For the example of clinical assessment, one can choose to deal with the business only in terms of services. Aligning services above processes is supported by the notion of Business Services Oriented Architecture (BSOA) [46]. It is argued that outcome is a concern taken before addressing how it should be produced.

Even though processes are mapped to services in PROSERVE, this thesis does not consider a process and a service to be semantically the same. A process outlines activities sequentially, while service considers value transactions between entities. While processes are responsible for creating values that are exchanged in services, services are responsible for creating values in processes. This dependency suggests that services and processes should co-exist in a common model in a way so that a service can be contained in a process and a process can be contained in a service.

A dissemination of the service concept in business modelling can also contribute to making organizations more conscious about service orientation of the enterprise, which probably is a premise for succeeding with SOA as a business approach. By using the service term to describe a business, it may be cognitively easier to understand the correspondence between business and IT since one is using the same metaphor in both domains.

A common strategy for hospitals in Norway is to outsource supporting information services to IT³⁷. At the moment there is an initiative in South-Eastern Norway Regional Health Authority to establish process modelling as an artefact for business improvement and improved IT-delivery³⁸. Process-orientation may be a step for the hospitals toward a business flavoured SOA. This is the classic SOA approach; processes in the business layer and services in the IT layer. However, this thesis suggests that the organization also uses a business service artefact in the business layer to align *what* is delivered, by *whom* and *how*. By including a business service artefact, the hospital has a means to describe contracts between entities in the organization and to IT in order to place responsibilities for service deliveries.

³⁷ Source: Strategic goals for South-Eastern Norway Regional Health Authority, strategic goals for Rikshospitalet University Hospital and strategic goals for Ullevål University Hospital.

³⁸ Source: Knut Hellwege (special IT-advisor for South-Eastern Norway Regional Health Authority)

A problem encountered in this thesis is the semantic confusion of terms around SOA. For instance, the term business service can both refer to an actual software service and a service independently it is realization. In addition, every combination of the words business, enterprise, software and service is already used both academically and commercially, making it hard to create unique terms for the different phenomena. This was the main motivation for creating a taxonomy for services in section 5.1.1.

Managing two concerns is difficult. The individual complexity of the business and software domains makes them hard to merge mentally without favouring one of them. SOA has emerged from the IT-industry and has evolved from a software engineering way of thinking. Even though SOA has recognized the need to couple the paradigm better to the business, e.g. through business process management, it may still be argued that this is driven by technology and therefore also limited by technology. It is believed that SOA should try to capture the source motivation of what has to be delivered, that is the main stakeholders' definition of a service rather than IT-people's potentially technology-constrained view upon services.

Operations

The term service operation was used in the literature that was reviewed [19, 26]. However, the meaning of it was not well explained or tied to semantics of the underlying modelling language. The notion of operation can be used both to describe what to do and what to deliver. These are different concerns. In PROSERVE, these concerns are divided in *ServiceFulfilmentContracts* and *ServiceDeliveryContract* as *Operations* and *ServiceOperations*, respectively. It may, however, be questioned whether “*operation*” is a good metaphor to describe these concerns in the business domain.

ServiceOperation should be able to both describe what it delivers and what it needs from the requester in order to deliver. The challenge is to do it in a way that promotes business thinking rather than software thinking. At the same time it is desirable that the information is structured so that software service artefacts can be derived.

In UPMS, a capability is defined as functionality that addresses a need [26]. It is in retrospect assessed as a better abstraction to describe service delivery than service operation, since it is regarded to be semantically closer to the notion of delivery.

8.7. Summary

In this chapter, the solution was evaluated with regards to its requirements, usability and its success criteria. While one of the requirements was only partially fulfilled, all of the success criteria were fulfilled. It was then possible to validate the hypothesis. The discussion in this chapter covers alignment of the research with existing solutions, its context of use and critique of PROSERVE and related state-of-the-art tool frameworks.

Chapter 9. Conclusion and Future Work

This chapter concludes this thesis by summarizing the work conducted, stating contributions and providing an outlook for future research work. First, the work is summarized in section 9.1. Then claimed contributions are outlined in section 9.2. This thesis finishes with section 9.3 by proposing future work based on the work carried out.

9.1. Conclusion

The main objective for this thesis was to improve modelling artefacts by providing a technical foundation for service description capturing with context support. A solution was proposed through creation of a metamodel that enables conceptual business and software services to co-exist in a common model with contexts separated (context-views), automatic process-driven business service identification, and a means to selectively abstract presentation of services in diagrams (context-overviews). An evaluation of the success criteria in section 8.5 shows that predictions were fulfilled. As a consequence, the hypothesis of this thesis was strengthened:

A business service modelling tool based on a service context scheme and a process-driven service identification will provide a means for presenting services in the context they belong to, so that SOA can be better coupled to business models through the use of the service metaphor in both domains.

A service context scheme was created as a solution for context-views and interactions of services and requisitions based on participant types. This scheme was then embedded in a synthesized metamodel for service modelling by leveraging on existing frameworks, in particular UPMS. The metamodel and a notation were then realized as a tool. In order to put the service tool into a larger context of enterprise architecture, a solution for identification of business services from business process models was made. In order to support focus on details in the context they exist in (context-overview), a selective abstraction mechanism was realized as a service tree. This part of the work leverages in particular on Richard Li's EML. A cognitive dimension analysis showed that the service tree provides a good mechanism for finding and viewing related elements in the diagrams, but hidden dependencies need to be reduced. All requirements were fulfilled, except from one partial fulfilled requirement.

From the discussion in section 8.6, is suggested that a technology independent service metaphor should be disseminated better at the business level of organizations. Moreover, a further harmonization of service modelling and business process modelling is needed.

9.2. Claimed Contributions

The contributions in PROSERVE are delivered through the artefacts: service context scheme, business-to-service mapping rules and service tree. Their contributions can be divided into four categories:

- Automatic process-driven business service identification.
- Context-views.
- Context-overview.
- Context-interaction.

Automatic Process-driven Business Service Identification

A solution for how business services can be automatically identified from business process was created by leveraging on findings in the reviewed literature. After defining a set of transformation rules, a model transformation could be executed, and new services were derived as proven in the experiment. Both SOMA and COMET use the concept of service identification from business processes, but these methodologies have not defined explicit transformation rules. EML represents a secondary syntax for services, but it has no underlying service metamodel. It can then be claimed that the PROSERVE solution represent some degree of uniqueness with regards to the tools it was compared with.

However, the usefulness of the solution was questioned in the discussion. So, its primary contribution is regarded to be the mapping rules which can be used as analytical guidelines for business service identification and as a basis for further work on harmonization of the notions of processes and services.

Context-views

A context scheme based on participant types provides a solution for separating contexts in service modelling. With regards to the tools in the state-of-the-art evaluation, this contribution is unique. However, PROSERVE should also be compared with tools that are more oriented towards enterprise architecture in order to assess uniqueness.

Context-interaction

The service context scheme also provides rules for which types of participants that can interact through services, both within a view and across views. Moreover, service splitting was introduced as a concept for keeping human and electronic delivery of a common service concept divided. This provides a means for depicting different types of service interaction that is described by Froehle [33].

Context-overviews

The final contribution is selective abstraction of services through use of service trees in diagrams. The solution leverages on Richard Li's work on EML. In PROSERVE, horizontal trees are used to represent related artefacts, while vertical trees are used to represent sequences, i.e. service behaviour and service traceability. Compared to Richard Li's work, the tree artefact constructed in this thesis is more flexible with regards to which elements are used in tree construction, e.g. the user can chose to add service owners as a new root for the tree, but it is not imperative. Moreover, a resizing mechanism is created, i.e. resizing the root node results in other tree elements being set to the same size. This strengthens the support for overview.

Finally, this thesis produces input to the discussion on how to utilize the service metaphor in public service sector through a case study from healthcare.

9.3. Future Work

9.3.1. Completion of Current Work

Parts of the metamodel were not realized due to shortage of time. These include the metamodel entities *ServiceDeliveryContract*, *ServiceOperation*, *Collaborator* (partially realized) and *ServiceRole* (partially realized). Future work should include these entities, as

well as extending the metamodel and tool with other entities that naturally belong in service modelling, e.g. message.

The derivation and use of a *ServiceRole* needs to be explored more. It is used to capture traceability for services with regards to which lane they are derived from. Although it may be more useful to link *ServiceRoles* directly to *Participants*, this question should consider how an organizational model can be used by the service model.

In regards to derivation of services from BPMN, the types of gateways should also be derived because they represent important details for requirements. A future metamodel needs to include a solution for this.

Customer representation needs to be improved in a way that enables better descriptions of the transfer of values across the organizational border. The introduction of *Collaborators* is a step toward a solution, but more work is needed to analyze the interaction in order to come up with improved concepts that capture customer-oriented aspects.

The metamodel also needs to support changes and re-engineering. A mechanism for coupling new requisitions to existing services must be created. For an example, the inclusion of collaborator participants in the metamodel is not of much value before there is a mechanism to couple their requisitions with existing services. In order to support connection between existing artefacts it has to be decided how formal the contract should be and how *ServiceOperations* or capabilities are used.

The notation and layout needs further work. Minor effort has been put into the design of the shapes and needs to be changed so that they are more intuitive. In addition, the problem with crossing connectors should be reduced. An approach this can be to incorporate a forced directed graph algorithm that draws graphs in an aesthetically pleasant way [80].

9.3.2. A Health Care Challenge

Population extrapolation predicts that the proportion of elderly people in the population will increase dramatically in the near future [81]. The result is more people in need for care and a greater need for interaction between specialist health services, general practitioners and nursing [82]. Being large and complex organizations constantly faced with new demands, hospitals need to offer effective services with high quality at lower cost.

Health enterprises are under constant reorganization in order to increase production and cut expenses. A possible merger between Ullevål University Hospital and Rikshospitalet University Hospital represents major organizational challenges where ICT plays a vital role.

Appropriate use of ICT is important in order to succeed in the further development of the health and social sector [82]. At the same time, the full potential for ICT investments within the sector is far from being fully realized. According to the Norwegian Directorate for Health³⁹, successful use of ICT involves securing speedy and reliable exchange of information between collaborating partners in the sector. This involves seeing ICT development in relation to developments in organizations and work processes.

Outsourcing of human services to IT and to external partners, involves semantic interoperability challenges both vertically between business and IT and horizontally between different enterprises. Semantical problems include lack of comprehensiveness, inadequate

³⁹ <http://www.shdir.no/>

implementations, lack of coordination between verbal definitions of concepts and technical representation of the same concepts [82]. An approach for improving IT support in the context of health can be to support understanding, adaptability and control of health enterprises through definitions in business architectures.

In large Norwegian healthcare enterprises, like Ullevål University Hospital (UUS), service orientation as a scheme for improving the business is still in its infancy. A reason for this may be limited economical resources in the sector to invest in competency and to develop and implement platforms and frameworks needed. Another reason may also be that the notion of service orientation is not well supported by the core business since it may be regarded as an IT approach for IT people. Moreover, service orientation requires organizational changes in order to work.

Stakeholders and decision-makers in hospitals are mainly clinicians that are motivated by giving patients the best possible treatment. When clinicians ask for IT support for patient treatment, they do not often talk about what kind of services they want from IT, but they often demand specific functionality that they want extended from existing applications⁴⁰. Conceptual business planning of support for core business is therefore often skipped. This makes it harder to make IT ends meet with business ends so that IT can offer services that fulfil business needs in the right way. As a consequence, IT solutions may not be well aligned with the organizations strategy.

The idea of enterprises being organized in dynamic and loosely coupled networks where they both can offer services and collaborate through services is not new within health care. However, SOA and SOA related frameworks and methodologies have not managed to support the need for representing services at a business level that are independent of technology in a way that is useful for strategic decision-making, business design, solution decision-making and rapid software development when services are outsourced to IT.

Message integration is a common way of integrating systems in health care, e.g. by using the HL7⁴¹ standard. A business process language is used to orchestrate activities that have to be executed. An orchestration is triggered by an incoming message and the content of the message is typically manipulated and sent to a target system. A common approach suggested in SOA is to define a business process in a high level business process language, e.g. BPMN, and transform it into an executable business process language, e.g. BPEL. However, there are several problems associated with this approach:

- High detail level in model intended for business people
- Software concerns in business model
- Lack of traceability and synchronization to business models
- Lack of semantic support for inter-organizational service interoperability
- No re-engineering support
- No means to make contracts between organizational responsibilities and IT responsibilities.
- Demanding to deploy on workflow engine

⁴⁰ Own experience as system manager, system integrator and enterprise architect at Ullevål University Hospital.

⁴¹ Health Level 7: <http://www.hl7.org/>

The main challenges outlined above are addressed as a matter of service interoperability which is related to three interoperability levels (technical, semantical and organisational) as defined by the European Interoperability Framework (EIF)⁴².

9.3.3. Improved Technology Artefacts for Service Interoperability

Enterprise interoperability is a relatively recent concept, but gains increasing attention in the research communities. It describes a field of activity which aims to improve the manner in which enterprises, by means of information and communications technology, interoperate and collaborate with other enterprises and organisations in order to achieve their business goals. EU has established an Enterprise Interoperability Research Roadmap⁴³ that tries to identify a research convergent path through future European Enterprise Interoperability. Challenges in Norwegian health care may be linked to this work.

It is believed that dissemination of the service concept in core business as the mechanism that organizational entities receives or offers deliveries of value is a success factor for enterprise interoperability. However, limitations in current SOA frameworks need to be attacked in order to improve intra- and inter-organizational service interoperability. Although PROSERVE can enable stakeholders and decision-makers in hospitals to deal with services at a high level of abstraction, it needs to include other disciplines of business architecture so that business process concerns can be reconciled with other business concerns.

SERVE (SERvice-oriented adaptable Enterprises) was an initiative to integrate the business perspective with the IT-perspective through a model-driven and a service-oriented approach [83]. A proposition for further work is based on this proposal.

It is assumed that the service metaphor can be the concept that integrates the two perspectives, while MDA is the means to dynamically propagate content between them. There are several ways the PROSERVE contribution can be taken further to resolve the problems of integration of the perspectives:

- *Harmonization of business processes and services.*
- *Integration of business architecture in a business service metaphor.*
- *Translation of the business service metaphor into a software service metaphor.*
- *Semantic support vertically between business and IT and horizontally between enterprises.*

A current research need is to find a solution for how service ends in business can be merged with service ends in software. Business people should be able to specify what they want independently of software concerns. At the same time, business service specifications should be defined in a formalism that enables some degree of automated reuse of these in an IT-context in order to support rapid realization and semantic correlation. Finding a suitable degree of formalism is probably challenging task which require empirical research. It is assumed that business people prefer a low degree of formalism and that a high degree of formalism will lead to low compliance of the solution (See **Figure 74**).

⁴² IDABC, "European Interoperability Framework for Pan-European eGovernment Services, Version 1.0", IDABC, 2004.
<http://europa.eu.int/idabc/en/document/3761>

⁴³ M.-S. Li, R. Cabral, G. Doumeingts, and K. Popplewell, "Enterprise Interoperability Research Roadmap, Final Version, Version 4.0", July 2006

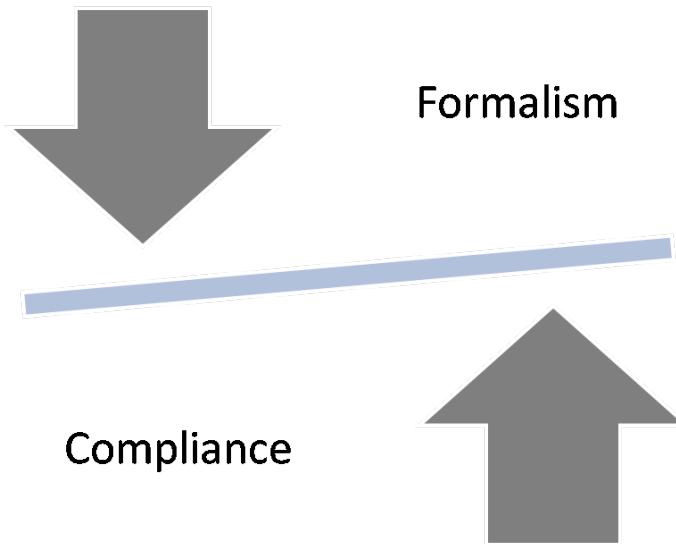


Figure 74: Trade-Offs - Determining the Degree of Automation.

Research needs to be performed in this area to provide tools that link business modelling with software system modelling, and to embed support for semantic service interoperability in the development process. It is suggested that the need for a solution can be met by building on some of the concepts in PROSERVE. A solution may consist of a suit of methodology frameworks and tools called SERVE. These can be divided into three platforms:

1. BIZSERVE: The business platform
2. INTSERVE: The integration platform
3. SOFTSERVE: The software platform

SERVE aims at specifying mappings between services in order to achieve semantic service interoperability, where the resolution of semantic mismatches will be based on semantic annotation of service models with reference to ontology and concept models from the business architecture level.

9.3.3.1. BIZSERVE

A future solution for an integrated business service model for business architecture may be called BIZSERVE.

In the discussion, it was argued pro a further harmonization of business service modelling and business process modelling. An approach for improvement may be to seamlessly integrate the two disciplines into a common metamodel and tool. The existing solution for automated business service identification needs to be improved.

PROSERVE provides a simple approach for working with business processes and business services. However, as suggested in the discussion, the integration could be taken further, so that services and processes are better aligned. It is suggested that BIZSERVE attempts to bind all dimensions in enterprise architecture (EA) together. An integration will assemble concerns about the businesses' *what, how, why* and *who*.

There are several approaches for doing this. Some of these may be:

- Mapping of business models to a service model.
- One large EA service model that incorporates all dimension of EA.

Mapping of business models to a service model can enable service identification and derivation of service descriptions from business models by the use of model transformation. Bi-directional mappings can enable tracing to the source, which can be used both for analytical support such as impact reports and traceability that can be supported throughout the SERVE solution in order to propagate changes. A key topic may be to deal with redundancy and potential inconsistency between the different business models.

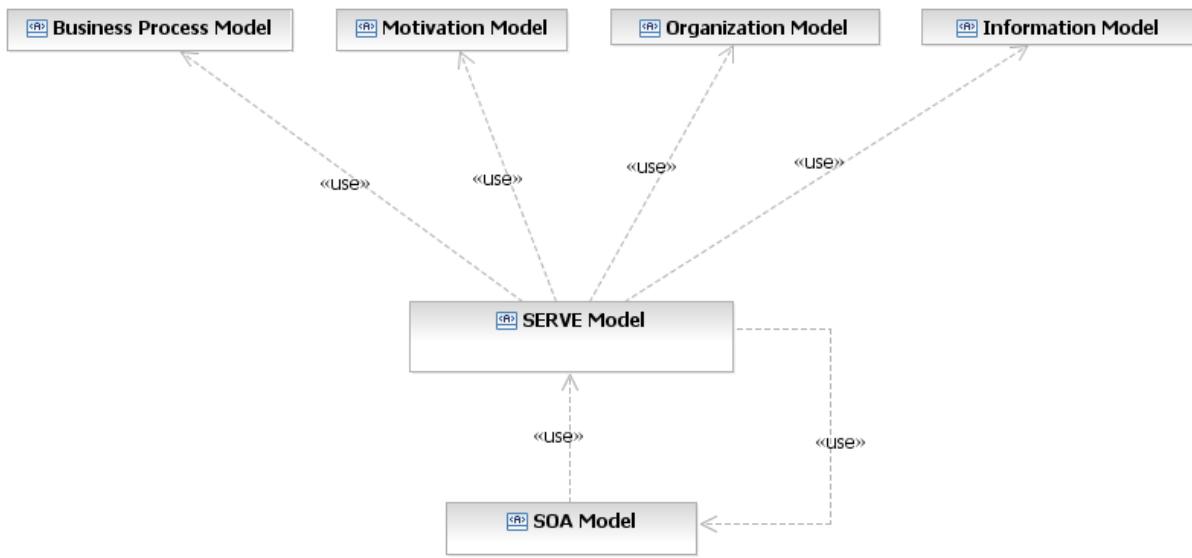


Figure 75: Vertical Mappings.

A large service model that can incorporate all dimensions of EA can be created. This will enable tighter integration of different concerns. However, this might lower flexibility since the model will constrain how the dimensions are represented. If the models are loosely coupled any type of business models can in theory be used to represent an EA dimension. On the other hand, the other solution raise challenges with regards to semantics, correlation and synchronization of the models.

BIZSERVE needs to improve operational artefacts in the PROSERVE metamodel, e.g. the notion of service operations. As discussed, a service operation is not well defined concept in SOA. The term operation may not be the right abstraction for describing specific service deliveries. It also needs to be reconciled with the notion of capabilities. Additional research is required.

Independent of the approach chosen, a significant analytic work needs to be done to reconcile the different concerns and semantics of artefacts. By using business standards, e.g. from OMG⁴⁴, this analytical work can be reduced. Future work should contribute to harmonization of standards for business modelling.

⁴⁴ Business Motivation Metamodel (BMM), Business Process Definition Metamodel (BPDM), Business Process Modeling Notation (BPMN), and Semantics of Business Vocabulary and Business Rules (SBVR).

9.3.3.2. INTSERVE

The objective of INTSERVE is to integrate the concerns of BIZERVE and SOFTSERVE. It aims at being a platform for negotiating requisitions from the business domain to service specifications in the software domain and vice versa. Three approaches can be supported: top-down, bottom-up and meet-in-the middle⁴⁵.

In a top-down approach, service requisitions are used to derive new service specifications that are propagated to SOFTSERVE for further refinement and realization. To do so, business requisitions and their related service contracts have to be translated into software service specifications. This may require additional formalization of the business definitions. A question raised in the work on PROSERVE is whether it is possible to integrate different concerns into a common metamodel. The indication from the evaluation is that this may be done in a conceptual EA context. However, a service model for software engineering that addresses software design issues and mixing these with business concepts in a common model may be problematic. On the other hand, a common metamodel can make integration easier and can be considered as an optional approach.

A bottom-up approach is assessed to be expensive in regards to resources. It is a time-consuming project to re-engineer existing IT-solutions into service models. Moreover, the existing software may not be service-friendly in the sense that they might be challenging to translate into a service representation. On the other hand, a service representation of these services will probably enable better control from the business side and decrease the time for making changes to them. In INTSERVE, re-engineered service capabilities can be exposed and used to derive requisitions that will be part of BIZSERVE. Even though this approach is not ideal [84], it may have to be done when better control of existing software solutions is needed and when forward-engineering is not an option.

The main purpose of a meet-in-in-the-middle approach is to facilitate reuse of software services. Business requisitions from BIZSERVE will be exposed and reconciled with software services in SOFTSERVE. As a result, both requisition and services may have to be changed in order to make agreements. Since all services and requisitions can be linked, impacts can be traced and analysed.

Renegotiation of service contracts can be done in INTSERVE in situations where design of business services is not suitable for software realization. Changes that are requested from SOFTSERVE can be proposed to BIZSERVE. On acceptance, changes have to be committed to all affected artefacts.

The realization of INTSERVE may not depend on its own service metamodel. It will, on the other hand, need strong support for correlation⁴⁶, traceability, synchronization and semantics to manage integration between BIZSERVE and SOFTSERVE.

9.3.3.3. SOFTSERVE

A target solution for software services specifications may be called SOFTSERVE. An end-to-end solution can be attempted to be achieved by extending the translation of services in INTSERVE with:

⁴⁵ Service identification in SOMA is based on these approaches.

⁴⁶ A project on a metamodel for correlation is in progress at the Department of Computer Science at the University of Auckland (ref PhD student Rainbow Cai).

- *Integration of software concerns in the software service metaphor, e.g. software design and software requirements that are not described in EA.*
- *Semantic annotations of Web Services and semantic mappings to business architecture.*
- *Deployment of software service specifications on executable platform.*

In INTSERVE, business logics should be translated into a software representation of the logics in SOFTSERVE model. SOFTSERVE meta-model needs to capture additional requirements for the service: All functional and non-functional requirements including performance, security and design requirements. In addition, it must be possible to design software service design.

Model descriptions that are derived from the business side need to be detailed out so that they can be transformed into executable software code and process execution code like BPEL or XLANG. This may be done via platform specific models (PSM). In order to achieve semantic service operability, semantic annotation of the code is attempted to be achieved. Semantic annotation can be mapped to definitions in a common ontology for services. The generated code then needs to be deployed in a software development environment for further refinement.

Model-Driven Interoperability (MDI) supports interoperability by utilizing principles and techniques from model-driven development, to abstract from platform specific models (PSM) to platform independent models (PIM). The semantic interoperability issues are resolved on the platform independent level, by various mappings between source and target models⁴⁷. SOFTSERVE metamodel is intended to be platform independent (PIM) in the sense that it does not rely on specific technologies. Additional platform specific models (PSM) may be used to capture rich models for rapid deployment on specific technology platforms. Research is needed to enable alignment of semantics between PIMs and PSMs.

SOFTSERVE will be based on current research on Model Driven Service Engineering (MDSE) which aims at combining Model Driven Engineering with Service Engineering. Service Engineering is an important European research topic, e.g. NESSI⁴⁸ and 3S⁴⁹. At the moment there are two European research projects that will be of interests for SOFTSERVE: The SHAPE⁵⁰ project which in September 2008 submitted the SoaML⁵¹ (Service oriented architecture Modeling Language) and SWING⁵² project that focuses on service composition for semantic web services.

9.3.4. SERVE for Ullevål University Hospital

The main goal for SERVE research is to resolve support for semantic service interoperability by the means of:

- Service-oriented EA (interdisciplinary use of service metaphor).

⁴⁷ Jean-Pierre Bourey & al. "Report on Model Driven Interoperability", INTEROP project, May 2007

⁴⁸ 19 Networked European Software & Services Initiative, <http://www.nessi-europe.com/Nessi/>

⁴⁹ 3S – Software & Services Strategy within the European Research Area,

ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/3s-project-story-revised_en.pdf

⁵⁰ Service-oriented Heterogeneous Architecture Platform Engineering, <http://www.shape-project.eu>

⁵¹ SoaML is the new name for UPMS. <http://www.omg.org/cgi-bin/doc?ad/08-08-04.pdf>

⁵² 23 Semantic Web services Interoperability for Geospatial decision making, <http://www.swing-project.org/>

- End-to-end service orientation (from business service definition to software service execution).
- Semantic coupling vertically and horizontally.

Another goal for the future research on services is to provide documented benefits for organizations. The work in this thesis is for instance purely conceptual and has not been tested in a real environment. For future research manifested in SERVE, it is vital to identify a solution's impact on an organization. It is therefore suggested that part of the research work should be to deploy a solution in an organization.

For Ullevål University Hospital (UUS), potential benefits of the research can be identified both in the EA axis to manage strategies, tactics and operational activities, and in the software domain in development and control of integration solutions within the hospital and across its organizational border. Support for semantic service interoperability can increase quality and speed of establishment of intra- and inter-organizational services.

Currently, UUS is involved in re-organization which aims at merging functions with other hospitals in the capital [4]. The size and complexity of this landscape make it hard to manage business, analyze implications and to keep up with the demand for integration solutions. Improved artefacts for service-oriented EA can provide better support for business planning and improve service interoperability.

A service-oriented EA will decrease the complexity of managing the services that the hospital produces. In a SERVE resolution of EA short-comings, the hospital can describe what it produces through services and how they are delivered through a service-driven approach. The service delegation mechanism described in PROSERVE can enable the hospital to define coarse-grained services that can be decomposed into supporting services in a divide-and-conquer strategy; making the business more manageable.

An assumption is that working service-oriented is easier for decision-making than working traditional process-oriented. For example stakeholders need not to consider the order of activities in early planning of business. From a top level view on the organization it is what is produced at which cost and implication that is most important. Responsibilities for production of services can be potentially be communicated and assigned in the organization in a service-driven way. However, a service-driven approach needs to be combined with process in order to deal with activities that require a specific ordering.

Embedding EA disciplines in service metaphors can enable the hospital to assemble all business concerns for what it produces under one umbrella. Today, the hospital has no means to trace related information for the services they need to provide. Moreover, the evaluation in this thesis of state-of-the art SOA solutions shows that there is lack of support for business services. Without support, assessments of changes may be incidental.

SERVE aims at finding a solution to trace dependencies so that affected objects of change can be identified. Another benefit of working with business service metaphors is that they may promote identification of opportunities to outsource service to IT and / or collaborators. This is a particular important feature since health care in Norway is in a continuously process of re-organization to cut expenses and maximize delivery.

A patient-centric focus is driving the health care business [79, 85]. The motivation can be made concrete and operational through definitions of services that support way of expressing involvement of patients in productions of these. In co-production of services there is a two-way flow values [6], i.e. patients contribute to increase value of services by providing input such as knowledge to the hospital. SERVE needs to create new or improved artefacts to support representation of exchange of values.

An important concept in the NIKT-report [79] is being able to identify services that are common across enterprises. By merging or outsourcing these services, they can be optimized and resources reduced. BIZSERVE can support the hospital in two ways. Firstly, by providing technology independent service definitions, the hospital has grounds for decision-making for outsourcing both to IT and to other businesses. Secondly, the hospital will have track on services that are affected if a service has to be changed, e.g. in situations where the service provider cannot conform exactly to the service specification or when services need to be changed in order to decrease expenses.

According to the report on service-oriented architecture from National-ICT (NIKT-report), expenses on point to point integrations in health care are at a very high level and increasing [79]. It is expected that the costs of maintaining and further development of these integrations will be higher than the benefits of new integrations. SERVE aims to resolve issues that hinder organizations like UUS to rapidly realize solutions for integration needs in a business-driven way. A solution for semantic service interoperability has the potential of increasing the production of the system integration team, increasing quality by increasing the conformity to business specifications and decreasing faults by using model-driven techniques.

Finally, SERVE intends to resolve how semantics in the business domain and the IT domain can be mapped in order to improve semantic service interoperability. Resolution of semantic problems is a key issue identified by the Norwegian Health Directorate for improving inter-organizational collaboration [81, 82].

From a research perspective, involving an organization such as UUS provides both an arena for proves-of-concepts and valuable experience in an academic field, i.e. EA, service-oriented architecture, model-driven development and model-driven interoperability. These fields still need to validate the usefulness of concepts in the real world. Conducting the research in an environment where the application of solutions can be tested continuously is assumed to contribute to optimization of the artefacts developed and maximize benefits for the hospital.

References

1. Paulson, L. D., *Service Science: A New Field for Today's Economy*. Computer, 2006. 38(8): p. 18--21.
2. Solheim, I. and Støhlen, K. *Technology Research Explained SINTEF A313*. 2007, SINTEF.
3. Denning, P. J., Comer, D. E., Gries, D., Michael, C. M., Allen, T., Turner, A. J., and Paul, R. Y., *Computing as a Discipline*. Communication of the ACM, 1989. 32(1): p. 9-23.
4. Spohrer, J., Maglio, P. P., Bailey, J., and Gruhl, D., *Steps Toward a Science of Service Systems*. Computer, 2007. 40(1): p. 71-77.
5. *Services Sciences, Management and Engineering*. 2008 [cited 2008 6th of August]; Available from: <http://www.research.ibm.com/ssme/>.
6. Vargo, S. L. and Lusch, R. F., *Evolving to a New Dominant Logic for Marketing*. Journal of Marketing, 2004. 68(1): p. 1-17.
7. Lee, J., *Model-driven Business Transformation and the Semantic Web*. Communication of the ACM, 2005. 48(12): p. 75-77.
8. OMG. *MDA Guide Version 1.0.1*. 2003. Available from: <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>.
9. He, X., Ma, Z., Shao, W., and Li, G., *A Metamodel for the Notation of Graphical Modeling Languages*. Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, 2007. 1: p. 219-224.
10. OMG. *MOF Model to Text Transformation Language RFP*. 2004. Available from: <http://www.omg.org/docs/ad/04-04-07.pdf>.
11. Greenfield, J. and Short, K., *Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools* 2004, Indianapolis, IN: Wiley Pub. 660.
12. OMG. *Meta Object Facility (MOF) Core Specification, v2.0*. 2006. Available from: <http://www.omg.org/cgi-bin/doc?formal/06-01-01.pdf>.
13. OMG. *Unified Modeling Language, Infrastructure, v2.1.2*. 2007. Available from: <http://www.omg.org/cgi-bin/doc?formal/07-11-04.pdf>.
14. OMG. *MOF Model to Text Transformation Language, v1.0*. 2008. Available from: <http://www.omg.org/docs/formal/08-01-16.pdf>.
15. Hahn, C., Madrigal-Mora, C., and Fischer, K., *Interoperability through a Platform-Independent Model for Agents*, in *Enterprise Interoperability II*. 2007, Springer. p. 195 - 206.
16. Ullberg, J., Lagerström, R., and Johnson, P., *Enterprise Architecture: A Service Interoperability Analysis Framework*, in *Enterprise Interoperability III: New Challenges and Industrial Approaches*, R.R. Kai Mertins, Keith Popplewell, Xiaofei Xu, Editor. 2008, Springer.
17. Lankhorst, M., *Enterprise Architecture at Work: Modelling, Communication, and Analysis*. 2005, Berlin ; New York: Springer.
18. The Open Group. *TOGAF*. 2008 [cited 2008 20th of October]; Available from: <http://www.opengroup.org/>.
19. Wahli, U., Ackerman, L., Bari, A. D., Hodgkinson, G., Kesterton, A., Olson, L., and Portier, B., *Building SOA Solutions Using the Rational SDP*. Redbooks. 2007: International Business Machines Corporation.
20. Havey, M., *Essential Business Process Modeling*. 2005, Sebastopol, CA, USA: O'Reilly Media, Inc.

21. WFMC. *Workflow Management Coalition Terminology and Glossary, Document Number WFMCTC-1011, Version 3.* 1999. Available from: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.
22. OMG. *Business Modeling Specifications.* 2008 [cited 2008 29th of August]; Available from: http://www.omg.org/technology/documents/br_pm_spec_catalog.htm.
23. Aalst, W. M. P. V. D., Beisiegel, M., Hee, K. M. V., Konig, D., and Stahl, C., *An SOA-based Architecture Framework.* International Journal of Business Process Integration and Management, 2007. 2(2): p. 91 - 101.
24. Spohrer, J., Vargo, S. L., Caswell, N., and Maglio, P. P. *The Service System Is the Basic Abstraction of Service Science,* in *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences.* 2008. Washington, DC, USA: IEEE Computer Society.
25. OASIS. *OASIS Reference Model for Service Oriented Architecture 1.0.* 2006, OASIS Available from: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>.
26. *UML Profile and Metamodel for Services (Revised Submission 2007-11-02).* 2007, IBM et al.
27. Berre, A.-J., *INF5120 Lecture Slides 25.02.2008.* 2008. Available from: http://www.uio.no/studier/emner/matnat/ifi/INF5120/v08/undervisningsmaterial/e/F06-2008_2502_abj.pdf.
28. Ullevål University Hospital. *Ullevål University Hospital.* 2008 [cited 2008 28th of September]; Available from: <http://www.ullevaal.no/>.
29. Abe, T., *Research Report No. 246: What is Service Science?* 2005. Available from: <http://jp.fujitsu.com/group/fri/downloads/en/economic/publications/report/2005/246.pdf>.
30. Rust, R. T. and Lemon, K. N., *E-Service and the Consumer.* International Journal of Electronic Commerce 2001. Volume 5(3): p. 85-101.
31. Roland, T. R. and Kannan, P. K., *E-service: A New Paradigm for Business in the Electronic Environment.* Communication of the ACM, 2003. 46(6): p. 36 - 42.
32. Santos, J., *E-service Quality: A Model of Virtual Service Quality Dimensions* Managing Service Quality 2003. 13(3): p. 233 - 246.
33. Froehle, C. M., *Service Personnel, Technology, and Their Interaction in Influencing Customer Satisfaction.* Decision Sciences, 2006. 37(1): p. 5 - 38.
34. Papazoglou, M. P. *Service-oriented Computing: Concepts, Characteristics and Directions,* in *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003 (WISE 2003).* 2003.
35. Sinderen, M. v., Almeida, J. P. A., Pires, L. F., and Quartel, D., *Designing Enterprise Applications Using Model-Driven Service-Oriented Architectures,* in *Enterprise Service Computing : From Concept to Deployment* R.G. Qiu, Editor. 2007, Hershey PA : Idea Group Pub., 2007. p. 132 - 155.
36. Qiu, R. G., *Information Technology as a Service,* in *Enterprise Service Computing : From Concept to Deployment* R.G. Qiu, Editor. 2007, Hershey PA : Idea Group Pub. p. 1 - 24.
37. Nayak, N. and Nigam, A. *Modeling Business Services for Implementing on Global Business Services Delivery Platforms,* in *The 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007 (CEC/ECC 2007)* 2007.
38. Zimmermann, O., Doubrovski, V., Grundler, J., and Hogg, K. *Service-oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned,* in *Companion to the 20th Annual ACM*

- SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications.* 2005. New York, NY, USA: ACM.
39. Papazoglou, M. P. and Heuvel, W.-J. V. D., *Service-oriented Design and Development Methodology*. International Journal of Web Engineering and Technology, 2006. 2(4): p. 412 - 442
40. Haines, M. N. *The Impact of Service-Oriented Application Development on Software Development Methodology*, in *40th Annual Hawaii International Conference on System Sciences, 2007 (HICSS 2007)*. 2007. p. 172b -172b.
41. OMG. *Business Process Modeling Notation (BPMN) Specification v. 1.1*. 2008. Available from: <http://www.omg.org/cgi-bin/doc?formal/08-01-17.pdf>.
42. Davies, R. and Brabänder, E., *The Event-driven Process Chain*, in *ARIS Design Platform - Getting Started with BPM*. 2007, Springer London. p. 105 - 125.
43. Lin, W.-B., *An Empirical of Service Quality Model from the Viewpoint of Management*. Expert Systems with Applications, 2007. 32(2): p. 364 - 375.
44. Maglio, P. P., Srinivasan, S., Kreulen, J. T., and Spohrer, J., *Service Systems, Service Scientistsm SSME, and Innovation*. Communications of the ACM, 2006. 49(7): p. 81 - 85.
45. Hepp, M. and Roman, D. *An Ontology Framework for Semantic Business Process Management*, in *Wirtschaftsinformatik 2007*. Karlsruhe. Available from: <http://www.heppnetz.de/files/hepp-roman-an-ontology-framework-for-SBPM-WI2007.pdf>.
46. White, S. A. and Miers, D., *BPMN Modeling and Reference Guide*. 2008, Lighthouse Point, Florida, USA: Future Strategies Inc.
47. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., Riegen, C. v., Schmidt, P., and Ivana Trickovic, *WS-BPEL Extension for People – BPEL4People*. 2005. Available from: <https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/com.sap.km.cm.docs/library/standards-and-open-source/standards/Languages%20for%20Defining%20Business%20Semantics/WS-BPEL%20Extension%20for%20People.pdf>.
48. OASIS. *OASIS: Advancing Open Standards for the Global Information Society*. 2008 [cited 2008 10th of October]; Available from: <http://www.oasis-open.org/news/oasis-news-2008-02-14.php>.
49. Li, L., Hosking, J., and Grundy, J. *Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-based Displays*, in *VLHCC '07: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. 2007. Washington, DC, USA: IEEE Computer Society.
50. Köth, O. and Minas, M., *Structure, Abstraction, and Direct Manipulation in Diagram Editors*, in *Diagrammatic Representation and Inference*. 2002, Springer Berlin / Heidelberg. p. 290 - 304.
51. France, R. and Rumpe, B. *Model-driven Development of Complex Software: A Research Roadmap*, in *FOSE '07: 2007 Future of Software Engineering*. 2007. Washington, DC, USA: IEEE Computer Society.
52. *Webster's Encyclopedic Unabridged Dictionary of the English Language*, B.G. Philip, Editor. 1996, Merriam-Webster: Springfield MA, USA.
53. SINTEF-ICT. *COMET - Component and Model-based Development Methodology*. 2007 [cited 2008 6th of September]; Available from: <http://www.modelbased.net/comet/>.
54. Grundy, J., Hosking, J., Huh, J., and Li, K. N.-L. *Marama: An Eclipse Meta-toolset for Generating Multi-view Environments*, in *ICSE '08: Proceedings of the 30th*

- International Conference on Software Engineering.* 2008. New York, NY, USA: ACM.
- 55. Papazoglou, M. P. and Georgakopoulos, D., *Service-oriented Computing*. Communications of the ACM, 2003. 46(10).
 - 56. Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice*. 1998, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 480.
 - 57. Eclipse Foundation. *Eclipse Graphical Modelling Framework*. [cited 2008 29th of October]; Available from: <http://www.eclipse.org/modeling/gmf/>.
 - 58. Arboleda, H., Cassallas, R., and Royer, J.-C. *Implementing an MDA Approach for Managing Variability in Product Line Construction Using the GMF and GME Frameworks*, in *5th Nordic Workshop on Model Driven Software Engineering*. 2007: Ronneby, Sweden.
 - 59. Moore, B., Dean, D., Gerber, A., Wagenknecht, G., and Vanderheyden, P., *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. Redbooks. 2004.
 - 60. Eclipse Foundation. *SOA Tools BPMN Modeler*. [cited 2008 23th of May]; Available from: <http://www.eclipse.org/stp/bpmn/>.
 - 61. Czarnecki, K. and Helsen, S. *Classification of Model Transformation Approaches*, in *online proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*. 2003.
 - 62. Eclipse Foundation. *ATLAS Transformation Language*. [cited 2008 23th of May]; Available from: <http://www.eclipse.org/m2m/atl/atlTransformations/>.
 - 63. Eclipse Foundation. *MOFScript*. [cited 2008 10th of September]; Available from: <http://www.eclipse.org/gmt/mofscript/doc/>.
 - 64. W3C. *XSL Transformations (XSLT) Version 1.0*. 1999 [cited 2008 23th of May]; Available from: <http://www.w3.org/TR/xslt>.
 - 65. Grønmo, R., Aagedal, J., Solberg, A., Belaunde, M., Rosenthal, P., Faugere, M., Ritter, T., and Born, M. *Evaluation of the QVT Merge Language Proposal*. Bericht Modelware/I1.2.1, MODELWARE IST Project 511731. 2005.
 - 66. Oldevik, J., Neple, T., Grønmo, R., Aagedal, J., and Berre, A.-J., *Toward Standardised Model to Text Transformations*, in *Model Driven Architecture – Foundations and Applications*. 2005, Springer Berlin / Heidelberg. p. 239 - 253.
 - 67. Oldevik, J., *MOFscript User Guide*. 2006. Available from: <http://www.eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide.pdf>.
 - 68. W3C. *Document Object Model (DOM)*. [cited 2008 14th of July]; Available from: <http://www.w3.org/DOM/>.
 - 69. Eclipse Foundation. *ATL Basic Examples and Patterns*. [cited 2008 10th of May]; Available from: http://www.eclipse.org/m2m/atl/basicExamples_Patterns/Tree2List/.
 - 70. Kutar, M., Britton, C., and Barker, T. *A Comparison of Empirical Study and Cognitive Dimensions Analysis in the Evaluation of UML Diagrams*, in *14th Workshop of the Psychology of Programming Interest Group, Brunel University, June 2002 (PPIG 14)*. 2002. Middlesex University: Psychology of Programming Interest Group.
 - 71. Green, T. R. G. and Petre, M., *Usability Analysis of Visual Programming Environments: A Cognitive Dimensions' Framework*. Journal of Visual Languages & Computing, 1996. 7(2): p. 131 - 174.
 - 72. Blackwell, A. and Green, T., *Notational Systems - the Cognitive Dimensions of Notations Framework*, in *HCI Models, Theories, and Frameworks*, J.M. Carroll, Editor. 2003, Morgan Kaufmann. p. 103 - 134.

73. Green, T. and Blackwell, A., *Cognitive Dimensions of Information Artefacts: A Tutorial*. 1998. Available from: <http://www.cl.cam.ac.uk/users/afb21/CognitiveDimensions/CDtutorial.pdf>.
74. Feldman, J., *What is a Visual Object?* Trends in Cognitive Sciences, 2003. 7(6): p. 252 - 256.
75. Joe McKendrick. *Why not sell SOA as ‘internal’ SaaS?* 2007 [cited 2008 12th of September]; Available from: <http://blogs.zdnet.com/service-oriented/?p=856>.
76. OMG. *Business Motivation Model (BMM) Specification, v1.0.* 2008. Available from: <http://www.omg.org/cgi-bin/doc?formal/08-08-02.pdf>.
77. OMG. *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS).* 2008. Available from: <http://www.omg.org/cgi-bin/doc?ad/08-08-04.pdf>.
78. Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*. 1999: Addison-Wesley Longman Publishing Co., Inc. 463.
79. Nasjonal-IKT. *Tjenestorientert Arkitektur i Spesialisthelsetjenesten, Versjon 1.0d.* 2008. Available from: http://www.nasjonalikt.no/Publikasjoner/Tjenestorientert_arkitektur_i_spesialisthelsetjenesten_hovedrapport_full_v1_0e.pdf.
80. Aaron Quigley. *What is a Force Directed Algorithm (Spring Algorithm)?* 2000 2003 [cited 2008 10 of September]; Available from: <http://www.cs.usyd.edu.au/~aquigley/3dfade/>.
81. *Respekt og Kvalitet - Rapport om Styrking av Spesialisthelsetjenester for Eldre (IS 1498).* 2007, Sosial- og Helsedirektoratet, Avdeling Sykehustjenester. Available from: http://www.shdir.no/vp/multimedia/archive/00029/IS-1498_Respekt_og_k_29579a.pdf.
82. *Te@mwork 2007.* 2006, Directorate for Health and Social Affairs. Available from: http://www.shdir.no/vp/multimedia/archive/00010/IS-1267_E_10176a.pdf.
83. SINTEF-ICT. *SERVE – SERvice-oriented Adaptable Enterprises - Research Proposal (Unpublished).* 2007.
84. Koehler, J., Hauser, R., Küster, J., Ryndina, K., Vanhatalo, J., and Wahler, M. *The Role of Visual Modeling and Model Transformations in Business-driven Development in The Fifth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2006).* 2006. Vienna, Austria: Elsevier Science B.V.
85. Helse Sør-Øst. *Mål 2008 for Helse Sør-Øst.* 2008 [cited 2008 10th of October]; Available from: http://www.helse-sorost.no/modules/module_123/proxy.asp?C=27&I=581&D=2&mids=.

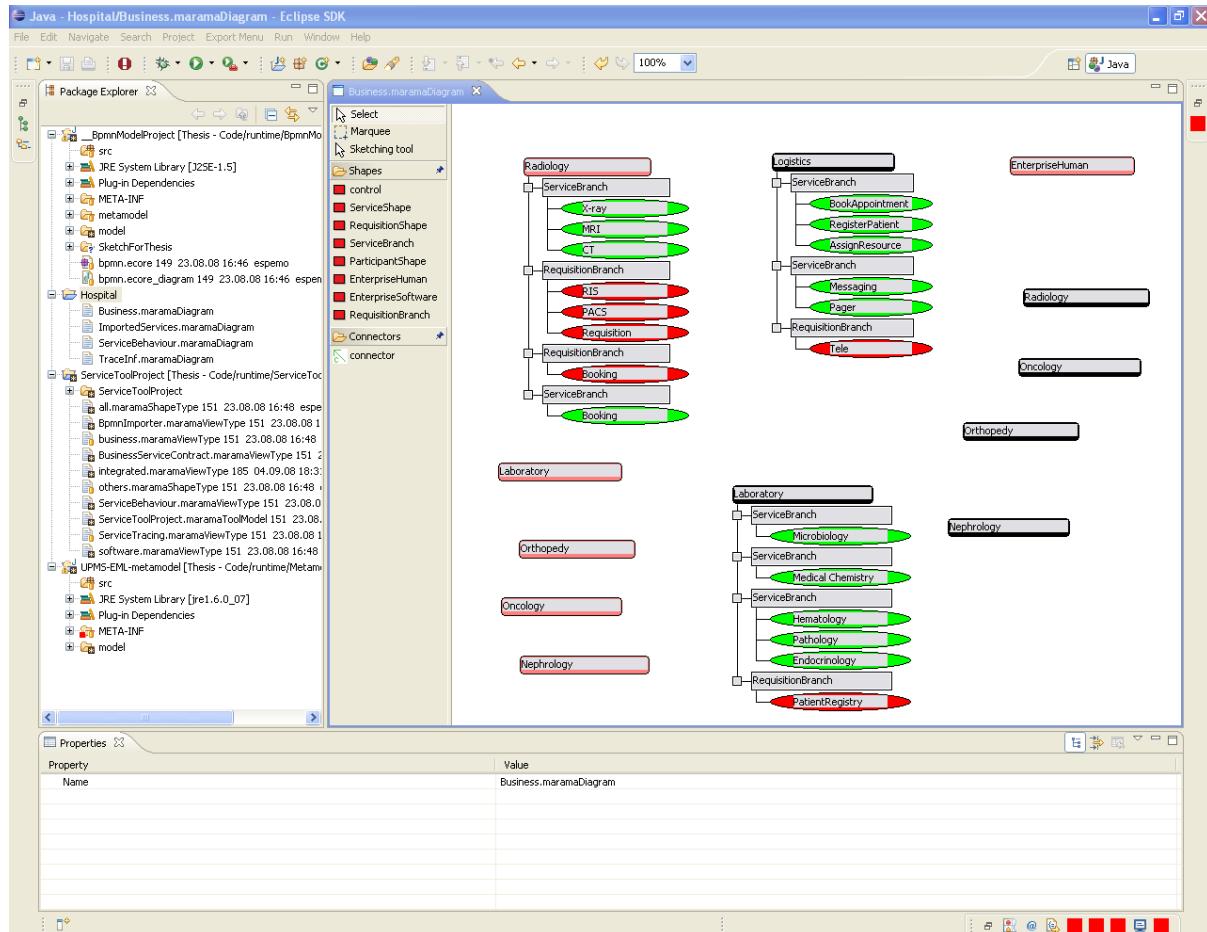
Appendix I: LIST OF ABBREVIATIONS

ADM	Architecture Developing Method
BaaS	Business as a Service
BPMN	Business Process Modeling Notation
BPM	Business Process Modelling
BSOA	Business Services Oriented Architecture
COMET	Component and Model –based Development Methodology
EA	Enterprise Architecture
EMF	Eclipse Modeling Framework
EML	Enterprise Modelling Language
GEF	Graphical Editing Framework
GMF	Graphical Modeling Framework
MDA	Model-Driven Architecture
MOF	Meta Object Facility
OSOAR	OASIS SOA Reference Model
PIM	Platform-Independent Model
PSM	Platform-Specific Model
SaaS	Software as a Service
SERVE	SERVice-oriented adaptable Enterprises
SOA	Service-Oriented Architecture
SoaML	Service-oriented architecture Modeling Language
SOMA	Service-Oriented Modeling and Architecture
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language
UPMS	UML Profile and Metamodel for Services
WS	Web Service
OMG	Object Management Group

Appendix II: PROSERVE TOOL INTERFACE

Business View

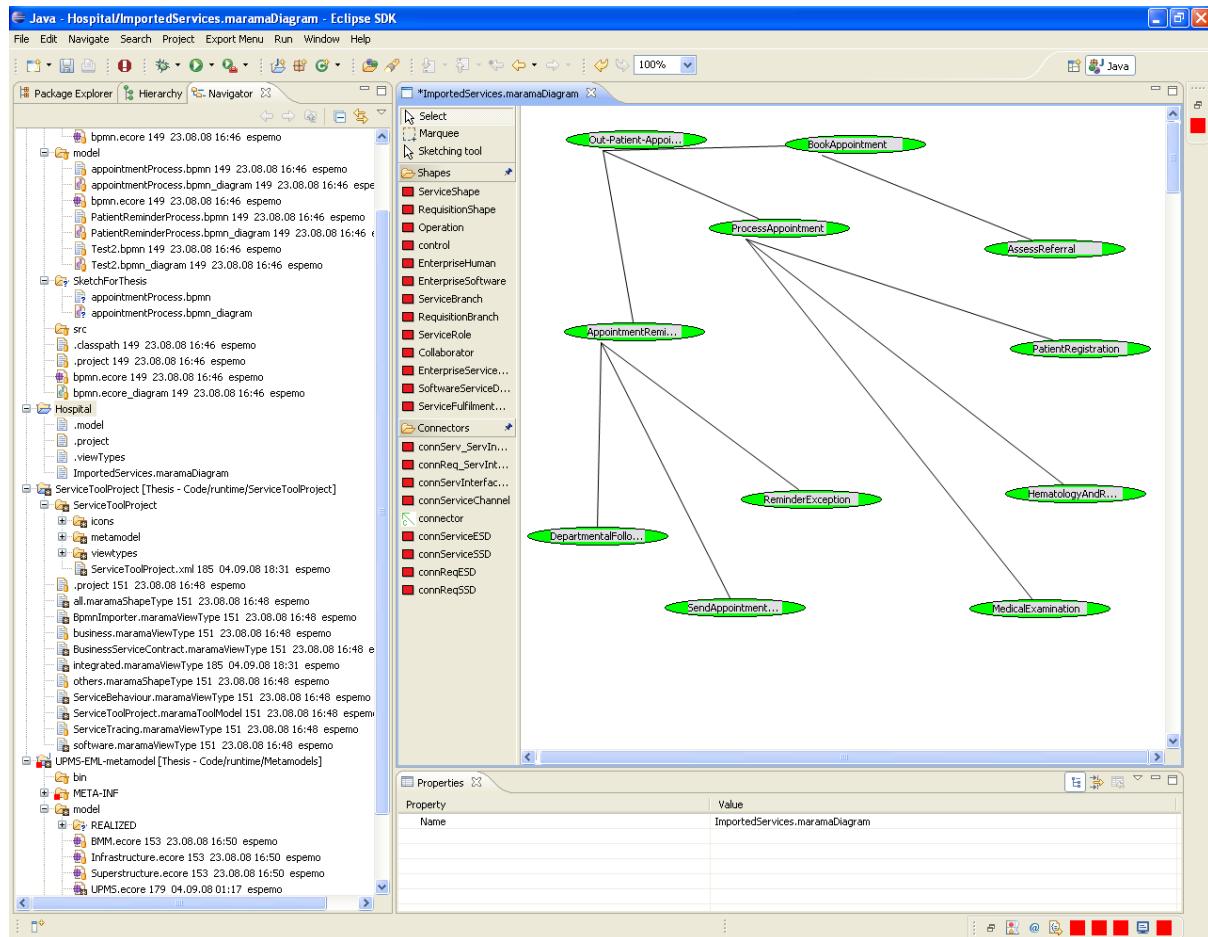
Diagrams in the business view present services definitions that are owned by EnterpriseHuman or EnterpriseSoftware.



Appendix Figure 1: Business Service Diagram.

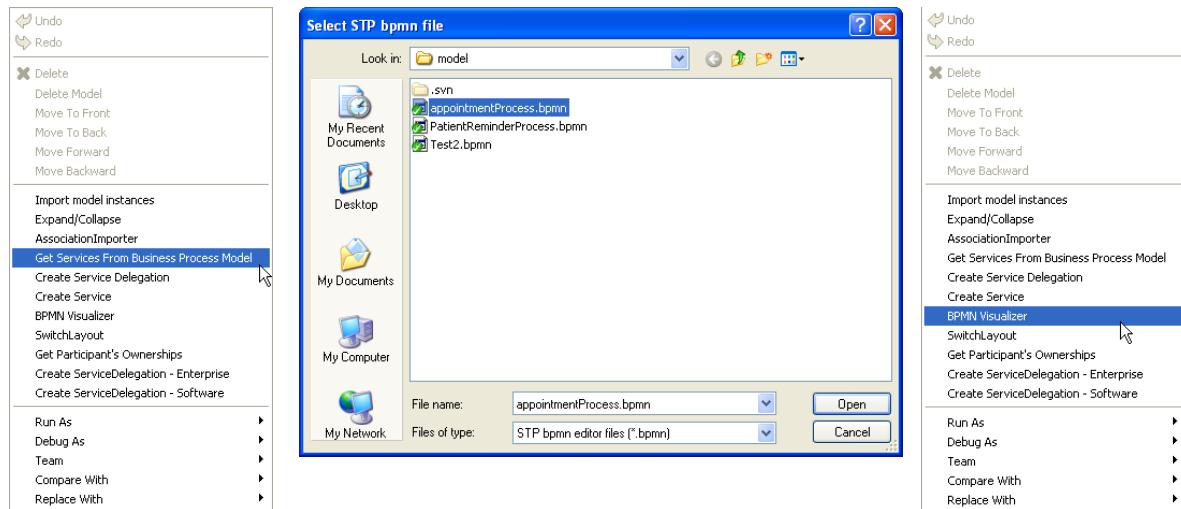
The Business Service Diagram provides an overview of service definitions in the business domain. Service Trees are used to offer a selective abstraction mechanism.

Appendix II



Appendix Figure 2: Business Service Contract Diagram.

The Business Service Diagram provides functionality for process-driven business service identification from BPMN models.



Appendix Figure 3: Identification of Services.

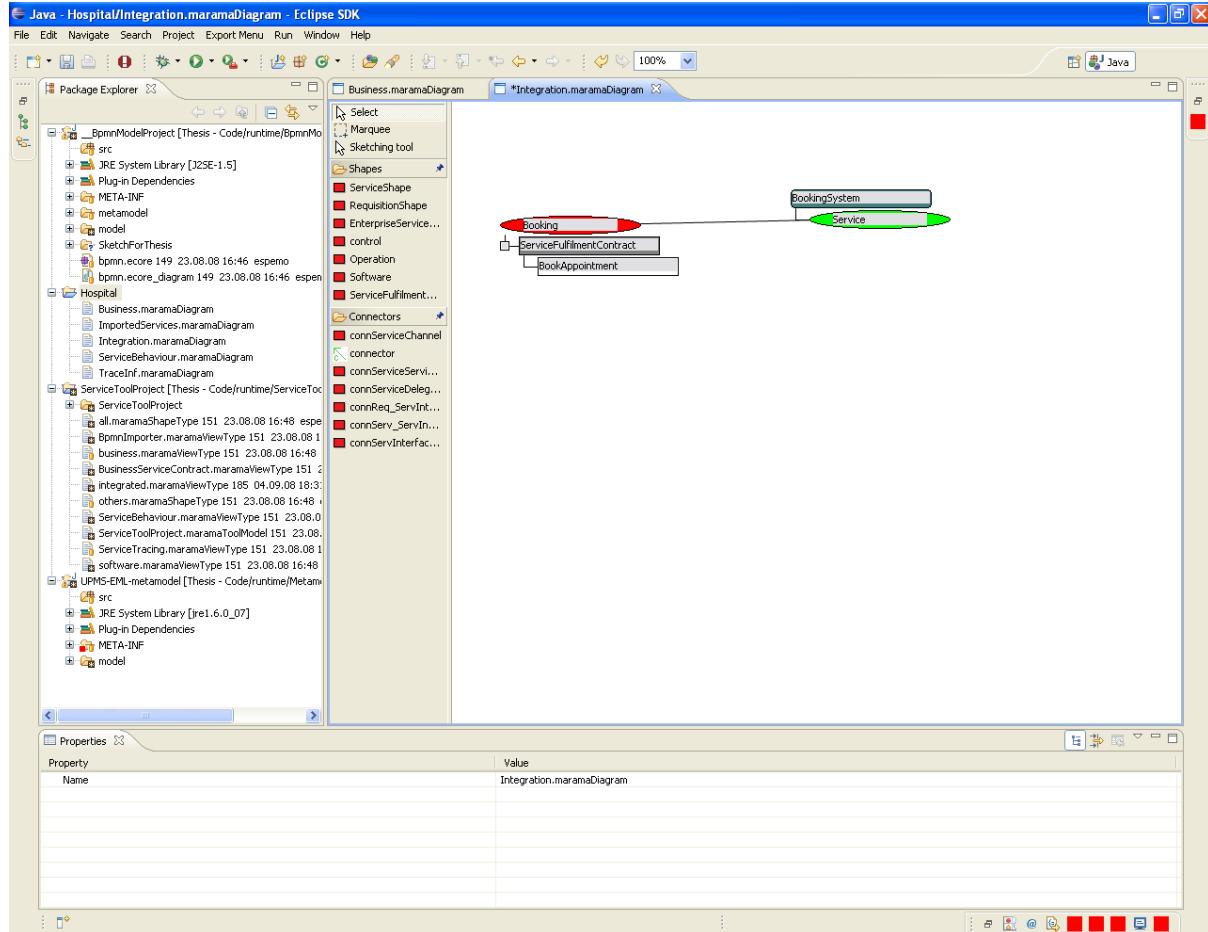
Services are derived from the business model by invoking the transformer from the context menu by selecting “*Get Services from Business Process Model*”. A wizard guides the user.

Appendix II

After the model has been populated, the user can chose to present the imported services in the diagram, by selecting “*BPMN Visualizer*”.

Integration View

In the Integration Diagram requisitions owned by EnterpriseSoftware can be imported and services to the software view can be derived.

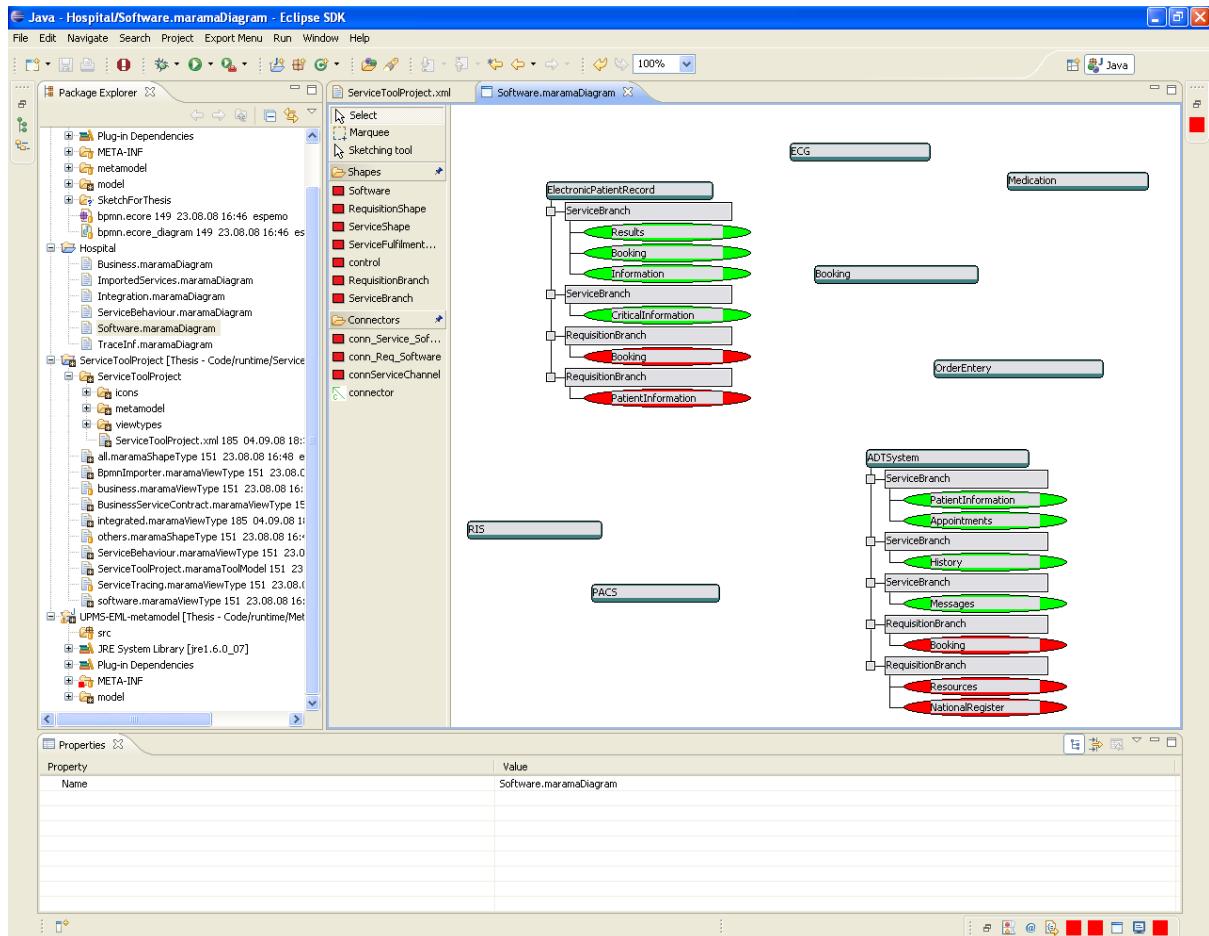


Appendix Figure 4: Integration Diagram

The user event handler “*Import model instances*” is used to import requisitions in the diagram. A *ServiceFulfilmentContract* have to be added in order to derive a new service. By using the user event handler “*Create Service*”, the new service and its owner are created automatically.

Software View

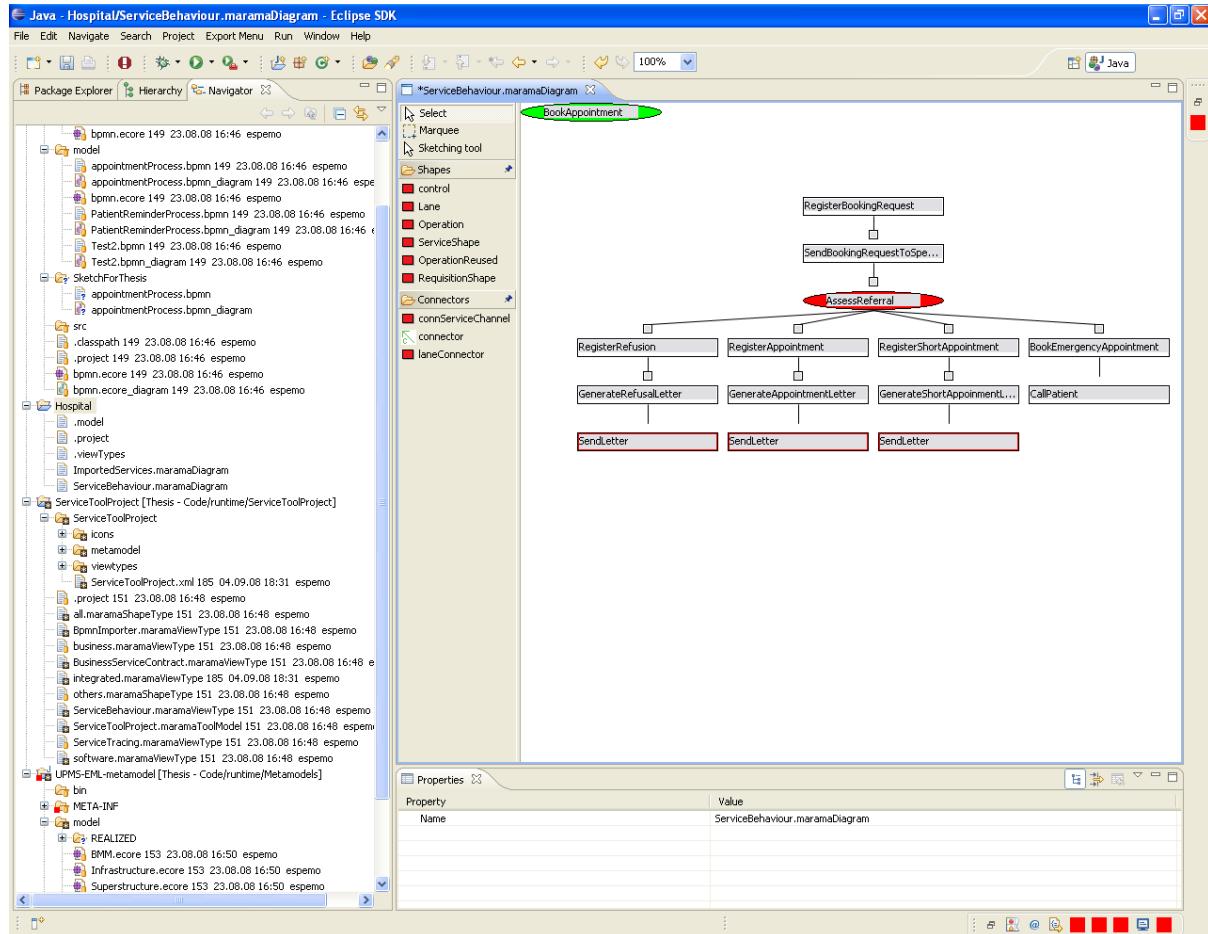
The Software Service Diagram can display services that are owned by Software.



Appendix Figure 5: Software Service Diagram.

All Views

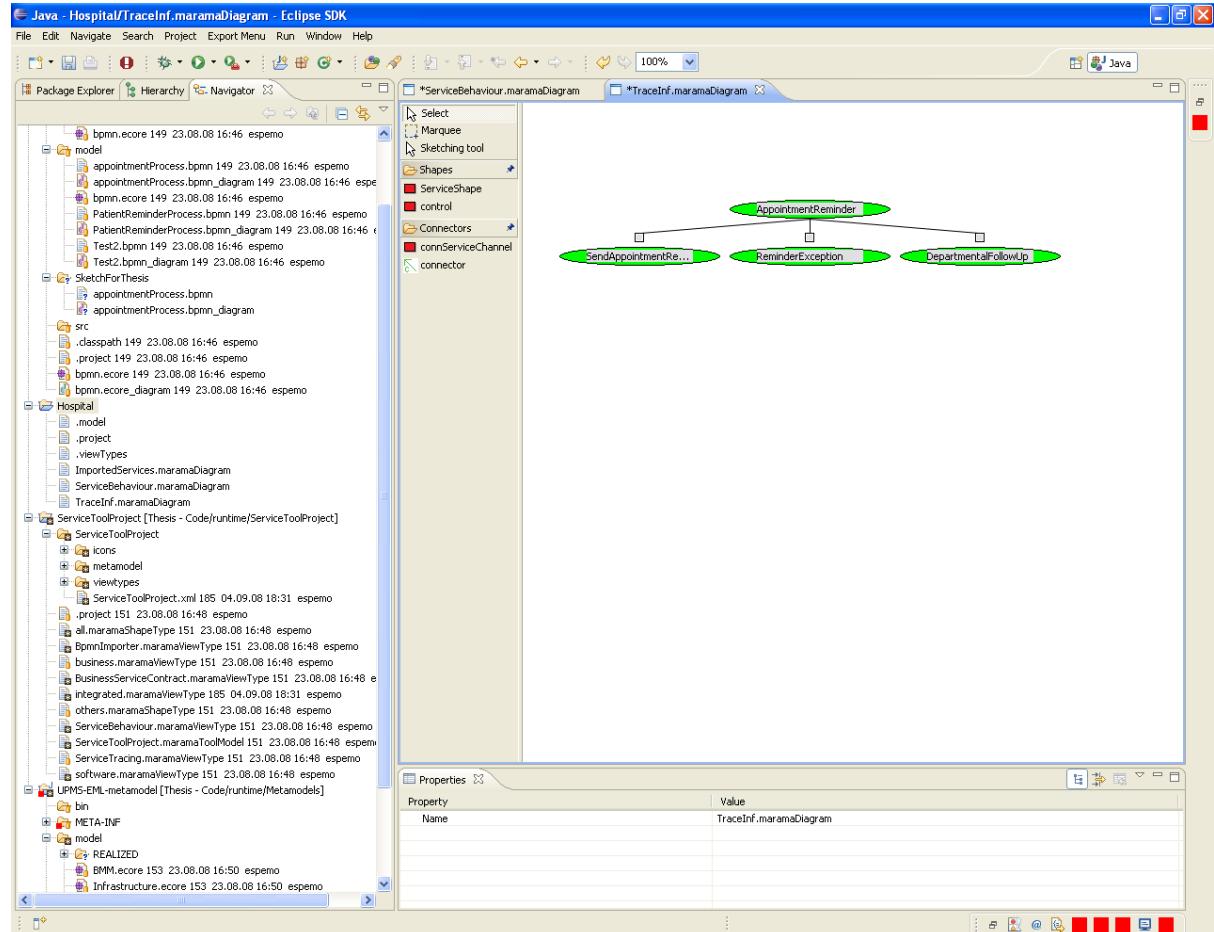
The Service Behaviour Diagrams can display the ordering of operations in *ServiceFulfilmentContracts* and dependencies to other services through representations of requisitions. A vertical tree is used to present service behaviour.



Appendix Figure 6: Service Behaviour Diagram.

Appendix II

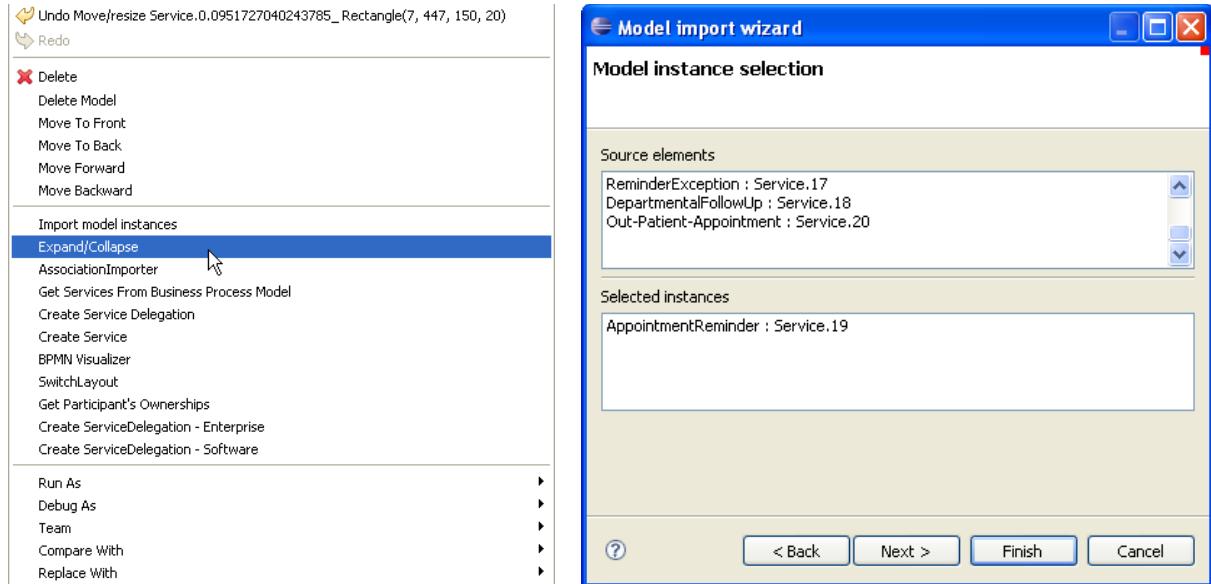
A Service Tracing diagram can display service dependencies as a vertical tree. The trace is invoked from the context menu. Both a bottom-up and a top-down trace are supported.



Appendix Figure 7: Service Tracing.

User Event Handlers

User event handlers can be invoked from a context menu in PROSERVE diagrams. By selecting a user event handler, a wizard can be triggered that guides the user through a few steps.



Appendix Figure 8: Context menu and Wizard.

Appendix III: PROSERVE DIAGRAMS

Business service diagram:

PROSERVE Model Entity	Remarks
EnterpriseHuman	Tree-node. Can contain Service Branches and Requisition Branches
EnterpriseSoftware	Tree-node. Can contain Service Branches and Requisition Branches
Service	Tree-node with capability of being aggregated in Service Branches.
Requisition	Tree-node with capability of being aggregated in Requisition Branches.
ServiceBranch	Tree-node that aggregates services
RequisitionBranch	Tree-node that aggregates requisitions

Business Service Contract Diagram:

PROSERVE Model Entity	Remarks
EnterpriseHuman	Tree-node. Can contain Service Branches, Requisition Branches, Services and Requisitions
EnterpriseSoftware	Tree-node. Can contain Service Branches, Requisition Branches, Services and Requisitions
Service	Tree-node with capability of being aggregated in Service Branches. Can contain EnterpriseServiceDelegation.
Requisition	Tree-node with capability of being aggregated in Requisition Branches. Can contain ServiceFulfilmentContract
ServiceFulfilmentContract	Tree-node with capability of being aggregated in Requisitions
Operations	Tree-node with capability of being aggregated in ServiceFulfilmentContracts.
EnterpriseServiceDelegation	Tree-node with capability of being aggregated in Services

Integration Service Contract Diagram:

PROSERVE Model Entity	Remarks
EnterpriseSoftware	Tree-node. Can contain Service Branches, Requisition Branches and Requisitions.
Software	Tree-node. Can contain Service Branches Requisition Branches and Services.
Service	Tree-node with capability of being aggregated in Service Branches and EnterpriseSoftware
Requisition	Tree-node with capability of being aggregated in Requisition Branches, EnterpriseSoftware and Software.
Service Branch	Tree-node that aggregates services.
Requisition Branch	Tree-node that aggregates requisitions.

Appendix III

Software Service Diagram:

PROSERVE Model Entity	Remarks
Software	Tree-node. Can contain Service Branches and Requisition Branches
Service	Tree-node with capability of being aggregated in Service Branches.
Requisition	Tree-node with capability of being aggregated in Requisition Branches.
Service Branch	Tree-node that aggregates services
Requisition Branch	Tree-node that aggregates requisitions

Service Chain Diagram:

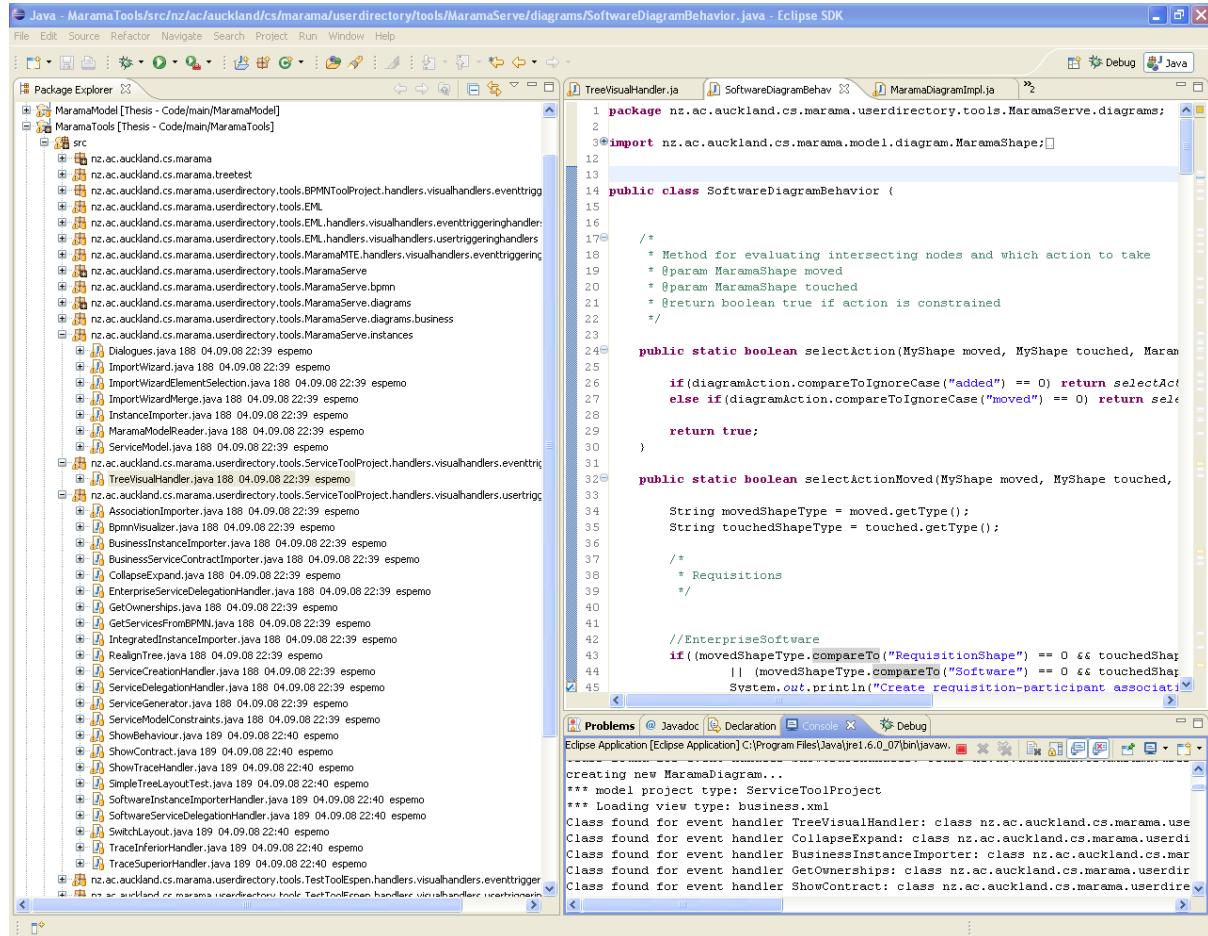
PROSERVE Model Entity	Remarks
Enterprise Human	Tree-node.
Enterprise Software	Tree-node.
Software	Tree-node.
Service	Tree-node.
Requisition	Tree-node.

Service Behaviour Diagram:

Service Model Entity	Remarks
Operation	Tree-node
Requisition	Tree-node

Appendix IV: REFERENCE TO DELIVERED TOOL ARTEFACT

This appendix was intended for listing of the source code for PROSERVE. The artefacts were generated in the Eclipse environment:



However, the size of it makes it unpractical to present on paper. The code will therefore be made available upon direct request to the author via e-mail. Please contact:

Espen Moeller: moller.espen@gmail.com