Development of a new service-oriented modelling method for information systems analysis and design
Prima Gustiené

Development of a new service-oriented modelling method for information systems analysis and design
To my family
Remigijus, Andrius and Tadas
Abstract

This thesis presents a new modelling method for information systems analysis and design, where the concept of service and the principles of service orientation are used for integrated modelling and reasoning about information systems architectures across organisational and technical systems boundaries. The concept of service enables cohesion of the intersubjective and objective modelling traditions by using a single type of diagram that facilitates detection of semantic inconsistency, incompleteness, ambiguity and discontinuity between the static and dynamic aspects of information systems specifications. The thesis is focused on three research topics, which are fundamental to the development of a new service-oriented modelling method. The first research topic concerns a pragmatic-driven specification of information systems. It clarifies answers to the research question: How can a conceptual modelling process be driven by pragmatic considerations? The second research topic provides a service-oriented modelling foundation for information systems analysis and design. It answers the research question: How can the concept of service be used explicitly for the analysis and design of information systems and how can the static and dynamic aspects of information systems specifications be integrated at the conceptual level? The third research topic presents transition principles to implementation-specific design and answers the research question: How can service-oriented conceptual representations be aligned with implementation-specific design?

The thesis contributes with a new knowledge to the area of conceptual modelling of information systems. The service-oriented modelling method consists of the modelling process, modelling language and techniques for the analysis and design of information systems on three levels of abstraction: pragmatic, semantic and syntactic. These three levels are necessary for a holistic understanding of enterprise architecture by stakeholders. The advantage of the service-oriented modelling method is that it can help to control traceability from information system design to original requirements. The method facilitates the semantic integration of the structural, behavioural and interactive aspects of information systems conceptual representations by using a single diagram type. The modelling language provides service-oriented constructs that are fundamental to building the major systems analysis patterns. The service-oriented modelling process contributes with seven steps of incremental design, which justifies various information systems components. The method provides the basis for a gradual and systematic way of modelling and an understanding of how pragmatic, semantic and logical information system requirements are linked together. The possibility to detect and eliminate undesirable characteristics of service-oriented diagrams can help to improve communication among stakeholders. Service-oriented specifications are computation-neutral and therefore they are more comprehensible for business analysis experts in comparison to implementation-specific graphical representations of information systems. Finally, this thesis presents the challenges for future research, one of which is the development of the automated tools for the alignment of business models with implementation-specific information systems specifications.
Acknowledgments

I am deeply grateful to all the people who have come together to help me to complete this thesis. I am very thankful to all my colleagues at the Department of Information Systems for their always friendly work atmosphere. It means a lot to me. Thank you for discussions and comments concerning my work.

I would like to thank my supervisor, Professor Anders G. Nilsson for his support and the discussions that we had while writing this thesis. I would also like to thank my co-supervisor Dr. Sten Carlsson for his encouragement and for being a co-author in one of the papers included in the thesis.

I would like to thank Dr. Fumio Negoro, President of the Institute of Software Development, Japan and Hamid Fujita, Professor at Iwate Prefectural University, Japan for providing me with the possibility to participate in the Lyee project and for their kind support in Japan. I would also like to thank HumanIT Centre for providing me with the possibility to participate in the project ‘Models and Methods of E-commerce’ that has been carried out at Karlstad University. I want to thank my colleague Irina Peltomaa, VTT Technical Research Centre of Finland for interesting research discussions during the conferences and for research cooperation.

Very many thanks to my mentor and the leader of enterprise and systems architecture design group at the Department of Information Systems at Karlstad University, Professor Remigijus Gustas for valuable discussions about modelling and for being a co-author in various research papers.

I would like to kindly acknowledge the effort of Dr. Jelena Zdravkovic, Stockholm University, for being the pre-opponent at my final seminar, for interesting discussions and valuable comments during the seminar.

I am very thankful to Michelle Cryer for proofreading my thesis. I am very grateful to Ann Dyrman for her kind help with EndNote tool for editing the bibliography.

I would like to thank my parents for their life-long support. You have always been an example and a source of inspiration for me. I am also grateful to my mother-in-law for her interest in my work and in believing me. I am very thankful to all my relatives and friends in Lithuania and in Sweden who supported me during all these years of hard work.

Finally, I would like to thank my family to whom I dedicate this work. I would like to express my gratitude to my beloved husband Remigijus and my two precious sons Andrius and Tadas for their love, encouragement, patience and everlasting support that has given new significance to my every accomplishment. I love you so much.

Karlstad, September 2010

Prima Gustiené
Contents

Part I.
Cover of Research 1

Chapter 1. General Observations Over the Thesis 3
1.1. Introduction 3
1.2. Problem Area 8
1.3. Research Focus 12
1.4. Goal of the Thesis, Research Topics and Questions 15
1.5.Disposition of the Thesis 20

Chapter 2. Research Approach and Methods 22
2.1. Research Approach 22
2.2. Research Methods 24
2.3. Philosophical Foundation of Research 27
2.4. Personal Background 31

Chapter 3. Related Work: Concepts and Background 33
3.1. Information Systems Analysis and Design 33
3.2. Enterprise Architecture 34
3.3. Enterprise Modelling 39
3.4. Model Driven Architecture and Modelling Levels 40
3.5. Conceptual Models and Modelling Language 44
3.6. Method Development 46
3.7. Conceptual Modelling in Information Systems Development 47
3.8. Concept of Service 49
3.9. Service in Service-Oriented Analysis and Design 51
3.10. Characteristic Features of Service-Oriented Analysis and Design 53
3.11. Desirable Characteristics of Service-Oriented Representations 55

Chapter 4. Service-Oriented Modelling Method: Modelling Process and Language 58
4.1. Modelling Process 58
   4.1.1. Pragmatic Dependencies 62
   4.1.2. From Pragmatic Specifications to Semantic Structure of Service 66
   4.1.3. From Actors to Syntactic Elements 68
4.2. Service-Oriented Modelling Language 70
   4.2.1. Interpretation of Modelling Concepts 71
   4.2.2. Notation of Static Dependencies 74
   4.3.3. Notation of Dynamic Dependencies 77

Chapter 5. Application of the Service-Oriented Modelling Method 80
5.1. Pragmatic-Driven Specification of Information Systems 82
   5.1.1. Goals, Problems and Opportunities 83
   5.1.2. Operations at Pragmatic Level 87
   5.1.3. Interplay Between Semantic and Pragmatic Representations 90
5.2. Service-Oriented Modelling Foundation for Information Systems

5.2.1. Core Elements and Perspectives of Service-Oriented Modelling 96
5.2.2. Intersubjective Perspective 98
5.2.3. Objective Perspective 100
5.2.4. Basic Semantic Constructs of Service-Oriented Modelling 101
5.2.5. Main Steps of Service-Oriented Modelling 105
5.2.6. Analysis Patterns for Service-Oriented Analysis and Design 114
   5.2.6.1. Service Interaction Pattern: Sequence 115
   5.2.6.2. Service Interaction Pattern: Synchronisation 116
   5.2.6.3. Service Interaction Pattern: Iteration 117
   5.2.6.4. Service Interaction Pattern: Selection 118
   5.2.6.5. Service Interaction Pattern: Search 120

5.3. Transition Principles to Implementation-Specific Design 121
   5.3.1. Extending Lyce Methodology Using Enterprise Approach 122
   5.3.2. Bridging Computation-Neutral Modelling to Lyce Diagrams 124
   5.3.3. From Conceptual Representations of Services to Object-Oriented Diagrams 133

Chapter 6. Contribution and Conclusions 146
   6.1. Publications and their Methodological Overview 146
   6.2. Summary of the Research Results Presented in the Publications 165
   6.3. Contributions of the Thesis 174
   6.4. Justification of Service-Oriented Modelling Method 178
   6.5. Implications for Different Target Groups 181
   6.6. Conclusions 182
   6.7. Future Research 185

Part II. Collection of Publications 189
   Paper 1. On a Problem of Ambiguity and Semantic Role Relativity in Conceptual Modelling 191
   Paper 2. Extending Lyce Methodology Using the Enterprise Modelling Approach 203
   Paper 3. On Desirable Qualities of Information System Specifications 223
   Paper 4. Enterprise Modelling Approach for Information System Engineering 233
   Paper 5. Towards the Enterprise Engineering Approach for Information System Modelling across Organisational and Technical System Boundaries 287
   Paper 6. Pragmatic-Driven Approach for Service-Oriented Analysis and Design 301
   Paper 7. Service-Oriented Foundation and Analysis Patterns for
Conceptual Modelling of Information Systems 327
Paper 8. Introducing Service Orientation into System Analysis and Design 347
Bibliography 399
List of Figures

Figure 1: Concepts concerning enterprise architecture .......................................................... 36
Figure 2: Heterogeneous architectural domains ...................................................................... 38
Figure 3: MDA framework and three modelling levels ........................................................... 41
Figure 4: Service as an interaction loop ................................................................................ 52
Figure 5: Architectural framework for service-oriented modelling ........................................ 59
Figure 6: Legend of the pragmatic dependencies .................................................................. 63
Figure 7: Goal decomposition .............................................................................................. 65
Figure 8: Opportunity as strength ......................................................................................... 66
Figure 9: Refinement of pragmatic entity .............................................................................. 67
Figure 10: Notation of components ....................................................................................... 68
Figure 11: Notation of interfaces .......................................................................................... 69
Figure 12: Interfaces between technical and organisational components .............................. 69
Figure 13: Relative interpretation of concepts ...................................................................... 72
Figure 14: Meta-model of concepts ....................................................................................... 72
Figure 15: Meta-model of relationship .................................................................................. 73
Figure 16: Classification dependency .................................................................................. 74
Figure 17: Inheritance dependency ....................................................................................... 75
Figure 18: Composition dependency .................................................................................... 75
Figure 19: Graphical representation of attribute dependency ............................................... 76
Figure 20: Graphical notation of dynamic dependencies ...................................................... 77
Figure 21: Interaction and transition dependencies .............................................................. 78
Figure 22: Presents an overview of architectural framework and research topics ................. 81
Figure 23: Opportunity as strength ....................................................................................... 84
Figure 24: Meta-model of opportunity .................................................................................. 84
Figure 25: Relativity between problem and opportunity ....................................................... 85
Figure 26: Relativity between goal and problem ................................................................... 86
Figure 27: Graphical representation of the union and intersection operations ....................... 88
Figure 28: Presents union operations .................................................................................... 88
Figure 29: Pragmatic decomposition ..................................................................................... 89
Figure 30: Presents pragmatic decomposition ...................................................................... 89
Figure 31: Pragmatic dependencies of Travel Agency ............................................................ 90
Figure 32: Decomposition of goal in terms of interaction loops ............................................. 92
Figure 33: Description of sub-goal ‘Reservation of Trip’ at semantic level .............................. 93
Figure 34: Refinement of sub-goal ‘Offer Discounts for Returning Customers’ ..................... 94
Figure 35: Two alternatives of goal ‘Sell a Trip’ .................................................................. 95
Figure 36: Actor dependencies ............................................................................................. 99
Figure 37: Example of semantic dependencies between actors .......................................... 100
Figure 38: Notation of information, physical and decision flows ........................................... 101
Figure 39: Representation of the semantic difference between two classes .......................... 102
Figure 40: Graphical notation of the attribute dependencies ................................................. 103
Figure 41: Graphical notation of the static dependencies ...................................................... 104
Figure 86: Class diagram ........................................................................................................... 144
Figure 87: Overview of the contribution .................................................................................. 175
List of Tables

Table 1: Design-science research guidelines and corresponding description .......... 22
Table 2: Scientific disciplines, which had an influence for the research study ........ 28
Table 3: Matrix of various representations of enterprise architecture .................... 37
Table 4: Characteristic features of SOAD ................................................................. 54
Table 5: The relation between research topics and publications .......................... 82
Table 6: Distinguished characteristics of publication number 1 .............................. 146
Table 7: Distinguished characteristics of publication number 2 .............................. 148
Table 8: Distinguished characteristics of publication number 3 .............................. 150
Table 9: Distinguished characteristics of publication number 4 .............................. 152
Table 10: Distinguished characteristics of publication number 5 ............................. 153
Table 11: Distinguished characteristics of publication number 6 ............................ 154
Table 12: Distinguished characteristics of publication number 7 ............................ 157
Table 13: Distinguished characteristics of publication number 8 ............................ 158
Table 14: Distinguished characteristics of publication number 9 ............................ 160
Table 15: Distinguished characteristics of publication number 10 .......................... 162
Table 16: Distinguished characteristics of publication number 11 .......................... 163
Part I

Cover of Research
Chapter 1

General Observations Over the Thesis

1.1. Introduction

Various models and methods are used to support the process of Information Systems (IS) analysis and design, but still after many years of practice, there are a lot of questions and unsolved problems that cause IS development projects to fail (Moody & Sindre, 2003). One of the reasons is that rapid changes in the business environment make it necessary to introduce new business design solutions, which should be effectively supported by computerized information systems. Such situations increase the complexity of information systems specifications and create difficulties for organisations in reaching their business objectives. Unprecedented rates of change in business processes and technological advances have made it more difficult for an information systems development team to be agile in responding to changing requirements (Lee & Xia, 2010). Business experts as well as system designers are not provided with the methods that support a systematic system development process across organisational and technical system boundaries. There lacks of a method that provides systematic guiding principles for aligning the overall IS design with respect to the enterprise’s goals (Kaindl, 1997). Traditional modelling methods do not provide guidelines over how pragmatic specifications that motivate business design can be linked to structural and dynamic aspects of IS specifications, which present data and business processes (Snoeck, 2003). The lack of a conceptual modelling method that helps to detect semantic integrity problems of IS specifications, creates a communication gap between business and Information Technology (IT) experts. Therefore, this thesis presents a new conceptual modelling method, which is challenging existing integration problems of IS specifications.

Concerning the development a new IS modelling method, some important themes in the area of information systems development have to be presented. This is necessary to motivate as well as explain what should be done and how this new method should be developed in order to facilitate IS reengineering agility and manage complexity. This introduction will start with a brief expose of the arguments concerning the complexity of information systems, and the importance of enterprise modelling and semantic integration for supporting the process of information systems analysis and design. This introduction will end with a short presentation of the service-oriented paradigm and with the presentation of its potential in solving IS integration problems. The introduction will be followed by the problem area subchapter. It is necessary for the clarification of existing problems in conventional modelling approaches. Chapter 1 will end with the presentation and description of the research focus, which provides the basis for the research topics of this thesis.
The most successful organisations are those that plan for necessary changes and are capable of implementing these changes (Lester, 2009). Such a situation requires, first of all, changes at the organisational level that lead to a permanent reorganisation and redesign of business processes. What is common to various kinds of computerized information systems is that they are all embedded into an organisation in order to improve the effectiveness and efficiency of organisations (Hevner et al., 2004). Information systems usually span across environmental, organisational and computerized information system boundaries. To know and understand information systems, it is indispensable to take into account the organisational environment, which is a wider system consisting of computerized and non-computerized system parts. The complexity of information systems causes difficulties in aligning the organisational subsystems with the IT subsystems. This is one of the most difficult problems facing today’s information systems analysis and design process (Maier & Rechtin, 2009).

Information systems analysis and design are two distinct activities in the process of system development. They are based on understanding the organisational objectives, structure and processes (Hoffer et al., 2004). These activities involve various stakeholders with different perspectives, purposes and backgrounds: the people who need the product, those who design and build it and those who deploy it. Understanding and mutual agreement among stakeholders is crucial for successful communication, which requires a common language. Enterprise Modelling (EM) activity (Babenko & Kinkova, 1999) is sometimes used to support early IS development phases in terms of goal models (Yu, 1997). EM quite often refers to the expression of enterprise knowledge (Stirna & Persson, 2009), which provides a holistic view of business goals, processes, rules, concepts, actors, resources and related implementation specific aspects. The notion of EM is often used interchangeably with business modelling (Nilsson et al., 1999), because business modelling also considers all EM dimensions such as processes, decisions and information (Grangel et al., 2007). The holistic understanding of enterprise requirements is critical for business experts, who determine organisational strategies. EM is important, because it helps in determining how information system components support a particular business activity and why this component is useful. Enterprise models also help to understand why business processes are performed and how they contribute to the objectives of the organisation.

EM and integration are two critical steps in IS analysis and design (Vernadat, 1996). EM is a major technique for dealing with complexity, which is inherent in the design and change of business processes (Biemans et al., 2001). This modelling process is important because it helps system developers to visualize, specify, construct and document static and dynamic aspects of IS specifications (Kruchten, 2006). Static aspects represent the structural characteristics of the system, emphasising the parts that make up a system. Dynamic aspects define the behavioural and interactive characteristics of the system, for example showing how the system behaves in response to external events. Models used to represent these aspects can be seen as the key to a successful integration process, as they focus on the interplay between
different aspects of the system. Integrated models should contribute to the process of validation and verification (Chester & Athwall, 2002), to the process of checking whether the system meets specifications and that it fulfils its intended purpose. This process is very important for reaching a better quality of system specifications (Lindland et al., 1994). The validation process usually refers back to the user's needs and is expressed by the question ‘Are you building the right thing?’ It requires the evidence that the final product accomplishes its intended requirements. It involves checking whether the system specifications are in accordance with the intentions of the people stated by goals (Edirisuriya, 2009). Verification is often an internal process expressed by a query ‘Are you building the thing right?’ To support the development of a systematic and integrated modelling method, my research is focused on three main topics, listed below.

• Pragmatic-driven specification of information systems.

A new service-oriented modelling method should help to analyse early IS requirements in terms of design goals (Ma et al., 2008; Mendes et al., 2001; Mylopoulos et al., 1999). In this thesis, the early stages of analysis are supported by pragmatic specification, which are important for the motivation of the ‘why’ dimension (Zachman, 1987). A pragmatic-driven, service-oriented modelling method should provide a way to analyse business intentions, which help to motivate static and dynamic aspects of IS specifications.

• Service-oriented modelling foundation for information systems analysis and design.

Traditional IS development methods project static and dynamic aspects into different types of graphical representations. This creates difficulties in reaching semantic integrity between various enterprise architecture dimensions. Static and dynamic aspects are complimentary. They describe the same artefact and therefore cannot be analysed in isolation. The isolation of information systems’ architectural views and dimensions creates difficulties in detecting inconsistencies and incompleteness of IS requirements (Moody & Sindre, 2003).

• Transition principles to implementation-specific design.

A service-oriented way of modelling should be computation-neutral, since the IS analysis process should not be influenced by any implementation solutions. This property is critical for bridging the communication gap between business and IT experts. Conceptual models are developed to support communication between stakeholders. Models and methods should facilitate unambiguous communication of design solutions among stakeholders and provide traceability of requirements. The models should help business experts and IS designers in the validation and verification of system specifications. In order to build a suitable IS, it is crucial that the terminology and representations used as a means of communication are unambiguous and understandable. An essential quality of modelling artefacts is their comprehensibility (Krogstie, 2006). If graphical representations used during the development process do not reflect the semantics of intended requirements, they can not serve as a basis for communicative purposes (Aranda et al., 2007). Therefore,
focus was also placed on the transition principles from computation-neutral modelling to implementation-specific design. It is also important to bridge this method to the conventional IS modelling approaches.

An integrated way of modelling both business and information technology (IT) system services is indispensable in the current reengineering practices (Lankhorst et al., 2004b). It is a prerequisite to developing a holistic understanding and to plan orderly transitional processes from the current to the new situation of an enterprise system. Achieving integration is problematic due to differences in architectural modelling methods (Steen et al., 2005). They lack a common language for shared knowledge (Falkenberg et al., 1996), which is important for bridging communication gaps among stakeholders by using shared business models and design methods (Nilsson et al., 1999). The current IS development tradition tends to draw attention away from strategic business modelling aspects and concentrates on the implementation-specific artefacts. Such a situation causes two problems. Firstly, the same implementation-dependent modelling methods are used in analysis and design phases. This increases a communication gap among stakeholders who are not IT experts. Secondly, an analysis of goals and other business-oriented aspects is left behind.

Service orientation is one of the fastest growing paradigms in IS development (Demirkan et al., 2008). It is not just a new paradigm for software engineering, but also a broader topic of Enterprise Architecture (EA) (Steen et al., 2005). The service-oriented paradigm has the potential to provide solutions to the IS specification integration problems mentioned above, because the phenomenon of service can be seen as a linkage between different enterprise modeling dimensions (Lankhorst, 2005). The purpose of this thesis is to develop a new modelling method for information system analysis and design, where the phenomenon of service and the principles of service orientation can be applied for IS analysis and design across organisational and technical system boundaries.

A service-oriented way of modelling should be based on the ontological principles of services and on a common understanding of the general structure of the service, which is not influenced by any implementation solutions. The ontological nature of the concept of service (Ferrario & Guarino, 2008) provides a new way of thinking, which is based on service interactions between organisational or technical system components. A service can be defined as a collection of ordered and purposeful interactions between actors that can be viewed as various human, organisational or technical components. This is important, since understanding of enterprise system architecture is reliant on knowing how different subsystems are interconnected. An enterprise system can be seen as a composition of interacting components, which are viewed as service requesters and service providers. An IS analysis and design process should have a much broader view to service orientation then Service-Oriented Architecture (SOA), which is an architectural approach for constructing complex software systems from a set of building blocks called services (Newcomer & Lomow, 2005).
Service-oriented modelling, should support the principle of separation of concerns (Jacobson & Ng, 2005), which is one of the guiding principles for managing IS complexity. The ability to apply the principle of separation of concerns helps to achieve the desired flexibility to reuse and change management. A service-oriented way of modelling enables explicit modelling of interaction flows (Gustas & Gustienė, 2009b) which is critical for identifying discontinuity in IS specifications and to comprehend the details of crosscutting concerns between different enterprise subsystems (Papazoglou, 2008). A new way of IS modelling should be based on service-oriented paradigm, which enables the integration of structural and dynamic aspects of IS conceptualizations.

Changes of service architecture need to be constantly captured, visualized and agreed upon. They are critical in aligning business and technical system design. Service orientation is an architectural style that helps to understand business processes as compositions of services (Erl, 2004) that can be changed by replacing individual services. Using a service-oriented way of modelling, information systems can be structurally visualized as evolving conceptualizations of service architectures (Gustas & Gustienė, 2009a). Service architecture is defined by business processes, which are compositions of organisational or technical services.

A new way of IS, based on the service-oriented paradigm, enables the integration of the structural and dynamic aspects of IS conceptualizations, which is critical for the integration of various architectural domains and to reach a holistic view of EA. To maintain a holistic view of enterprise architecture, and to understand the functionality of information systems, enterprise subsystems should be determined according to the design goals. A comprehensive way of business modelling cannot separate business-oriented details of services from structural, interaction and behavioural aspects. The modelling method should provide motivation for introducing software components and help to identify design problems in the early phases of IS analysis.

To achieve consensus and understanding among stakeholders, it is necessary to develop a method that provides a language, modelling process and modelling techniques for systematic IS analysis and design. The modelling process and techniques should help to detect semantic discrepancies of IS specifications on various levels of abstraction. Being pragmatic-driven, the modelling method should provide guiding principles for an incompleteness analysis of IS specifications with respect to goals. A new method should help business experts and system designers to define, visualize and assess various organisational changes by using a fully graphical approach to IS reengineering. This would in turn help to align business process design with organisational and technical components. Computation-neutral representations of service architectures can be used as guides, which support communication among stakeholders during information systems analysis and design process.
1.2. Problem Area

Information systems architectures are intrinsically complex engineering products that can be defined on various levels of abstraction and represented using different dimensions. These architectures are difficult to visualize across disparate modelling dimensions such as the ‘what’, ‘how’, ‘where’, ‘who’, ‘when’ and ‘why’ (Zachman, 1987). Integrating these dimensions is crucial, because they are projections of the same system. The traditional modelling methods do not provide effective support for business alignment with IT subsystems (Steen et al., 2005). For the same reason, EA is not easy to comprehend for the business experts who determine the organisational strategies. Various architectural descriptions are defined in the form of graphical representations, which are used during IS analysis and design process. These graphical representations are critical to understand how different system descriptions are analysed in relation to each other and how the requirements from different perspectives are perceived as a whole.

Large number of graphical representations used during the information systems development process makes it difficult to keep track of the modelling process and detect inconsistency between diagrams on different levels of abstraction. The design solutions, which affect different aspects of specifications, taken during the modelling process, should be traceable from one modelling dimension to another (Edirisuriya, 2009). To obtain value from the graphical representations they must be integrated and semantically correct. If graphical representations of the same artefact are not integrated, they may result in semantically incomplete, ambiguous, redundant and inconsistent specifications of IS. Low quality of IS specifications cause semantic problems of communication among stakeholders and contribute to the high failure rate of system development projects (Yoo et al., 2004).

There is still lack of an integrated architectural approaches that supports consistency among various views and perspectives (Lankhorst, 2005). Conventional approaches tend to draw attention away from the strategic business process modelling aspects (Singh, 2002) and concentrate on the implementation-dependent issues (Finkelstein, 2004). Generally, information systems specifications are not easy to validate for the simple reason that they span across organisational and technical system boundaries. These boundaries are not always clear as they are changing over time. Initial system requirements are often incomplete and ambiguous (Yu & Mylopoulos, 1998). If the requirements expressed in models are incomplete and inaccurate according to requirements, the implemented IS will not satisfy the users needs (Moody & Sindre, 2003). This may be regarded as one of the main causes of semantic problems of communication between stakeholders involved in the development process. Traditionally, the lack of understanding can be observed between two communities: business experts and IT experts. The problem is that business experts tend to consider technology issues as a subordinate aspect and IT
experts often consider that business modelling does not deserve much attention (Weske, 2007).

The most difficult part of IS modelling is to transform unclear requirements into a coherent, complete and consistent system specification of a desired information system (Gustas & Gustiené, 2004). The most difficult and important task of system developers is tying together separate structural and dynamic projections into a consistent and complete whole (Maier & Rechtin, 2009). The fundamental problem is that conventional information system development methods do not take into account certain important semantic interdependency types between the static and dynamic aspects, which are crucial for gluing the strategic, organisational and technical descriptions into an integrated representation (Gustas & Gustiené, 2009b).

Most traditional methodologies concentrate either on data, or process analysis (DeMarco, 1978), or object-oriented analysis (Rumbaugh et al., 1991), but do not deal explicitly with stakeholders’ intentions in terms of goals (Mylopoulos & Yu, 1998).

In the traditional modelling approaches, there is still a lack of attention on an ‘early-phase’ requirements analysis (Yu, 1997), which explains how the system would meet organisational goals and why it is needed. Despite this, the analysis and representation of a business strategy through goal modelling is crucial to achieve organisational alignment with IT subsystems. System specification in terms of goals is important as they are supposed to motivate and drive the overall IS analysis and design process (Gustas & Gustiené, 2008). Any engineering product should fulfil the needs of its potential stakeholders (Perrone et al., 2005a) and should be driven with respect to business goals. Goal modelling is important as it shows the strategic intent of doing business (Gordijn et al., 2000) and how various enterprise goals are related. Goal-oriented approaches allow the requirements to be analysed, refined and clarified through an incremental process (van Lamsweerde, 2004), which is important for step-by-step IS development. To the best of my knowledge, there is no modelling method with systematic guidance for an incremental way of modelling, starting from stakeholders’ intentions and leading to implementation-specific design.

It is not sufficient to analyse goals without relating them to static and dynamic aspects of IS conceptualizations. According to Kaindl (1997), the static and dynamic aspects of IS requirements should be related to the purpose of the system. Static aspects define the structural part of the system. They define ‘what’ is processed. Dynamic aspects imply interactive and behavioural aspects. ‘How’ and ‘when’ dimensions refer to behavioural aspects of the system, which define state changes of objects. ‘Who’ and ‘Where’ refer to interactive aspects, which define how the objects in a system cooperate to achieve necessary results. Usually, traditional IS methodologies are centred on modelling separate projections of IS in isolation. The lack of research on semantic integration of multiple diagrams (Kim et al., 2000) is not a new fact. The fundamental problem resides in the difficulty to integrate interactive, structural and behavioural aspects of information system specifications (Gustas, 1997), which is critical for understanding how different system descriptions are analysed in relation to each other and how the requirements from different perspectives are perceived as a whole.
Traditional modelling approaches for IS analysis and design, including Unified Modelling Language (UML) methods (OMG, 2009), are based on an object-oriented paradigm. Such approaches address the logical design of a system and facilitate the transition to code, but they are difficult to use for semantic integration and validation with domain experts. It is recognized that UML support for such tasks is quite vague (Perrone et al., 2005b). It adopts an implementation-oriented approach and lacks the integrated semantics of the static and dynamic aspects during the analysis and design phases. Object-oriented design (OOD) methods are used to model the structure and behaviour of a system, but do not explicitly provide its purpose. On the other hand, object behaviour analysis (Rubin & Goldberg, 1992) methods identify goals and objectives of the system, but do not show how they are related to the static and the dynamic aspects of IS specifications.

An object-oriented way of analysis has an implementation bias and usually follows the ‘bottom-up’ principle. This is not appropriate for business modelling, as it focuses on implementation-related details that are not important in the early stages of IS analysis. This contradicts the essence of requirements analysis that should only describe the problem domain, which is not influenced by any possible implementation solutions (Snoeck et al., 1999). According to Fowler (1997), analysis techniques are intended to be independent of technological solutions. Such independency increases the understanding of the problem domain. Most UML modelling primitives abstract from concrete implementation artefacts. Such modelling methods are more comprehensible and easier to use by software designers. Especially on the other hand, the technology-neutral representations of enterprise architecture (Finkelstein, 2004) can be used by non-technicians, who play a key role as semantic system integrators. It should be noted that service architecture should not prescribe any sort of implementation details. It must follow the basic conceptualization principle (van Griethuysen, 1982) by representing only computation-neutral aspects. This implies that a new developed method, including the process and modelling language, should not be influenced by any technology-oriented solutions at a very early analysis phase.

There are a number of methods that attempt to solve semantic integration problems. The Object Process Methodology (OPM) is an attempt to integrate behavioural and structural aspects of data. OPM puts into the foreground the modelling of static and dynamic relationships between objects (Dori, 2002). It emphasizes the difference between physical and informational objects. Physical objects can be subsystems, which are able to carry out actions and change informational objects that are understood as passive concepts. But the interactive flows in this methodology cannot be explicitly captured. Some IS modelling methods concentrate on the conceptualization of the external behaviour, and have the intersubjective tradition bias. This tradition puts into the foreground the modelling of the external behaviour of different actors (Dietz, 2006b). One such example is the ArchiMate (Lankhorst, 2005) modelling language, which is intended as an enterprise architecture modelling language for the definition of relationships between concepts in different domains. From the intersubjective point of view, service is a unit of
functionality, which is exposed to some environment. External behaviour helps to understand a usage aspect of autonomous subsystems, which help to define service architectures. Nevertheless, the ArchiMate language is quite weak in representing the interplay between service interactions and the associated changes in various classes of objects. Some approaches try to combine modelling techniques using formal mathematical and logical foundations for UML notation (Pons & Baum, 2000), but logics within a first-order formalism is still difficult to understand for people without mathematical background.

The analysis of the static and dynamic aspects of IS specifications in isolation creates fundamental difficulties in conceptual modelling of IS specifications. The consequence of analysing the static and dynamic aspects in isolation is that additional semantic modelling assurance procedures are necessary to establish integrity across multiple diagrams. Verification of semantic integrity between business processes and business data in such a situation becomes very difficult. Integrity problems imply semantic inconsistency and incompleteness of conceptual representations on various levels of abstraction. There is a lack of effective modelling methods that could help to detect these undesirable qualities of conceptual representations. These undesirable qualities of IS system specifications lead to semantic problems of communication among stakeholders. IS analysts use conceptual models to help different stakeholders to reach a shared understanding about the problem domain. The problem is that, very often, conceptual models are unclear and ambiguous (Burton-Jones & Weber, 2003). Such models cause problems in communicating design solutions and create difficulties in comprehending information systems as a whole.

The ambition of the service-oriented modelling method is to conceptualize organisational architecture in interplay with informational objects, which are created, modified or terminated by the technical components. A new developed service-oriented modelling method is intended to support communication among various types of stakeholders such as: owners, system designers and enterprise architects. It must provide help in achieving a better quality of IS conceptual representations such as unambiguity, independency, verifiability and traceability of requirements. Achieving these desirable characteristics is important for the following reasons:

• unambiguous requirement guaranties that interpretation in at least two different ways is impossible.
• independency should support the clear principle of the separation of concerns. Early requirements analysis is important for the separation of concerns of different stakeholders (Jacobson & Ng, 2005). Some concerns can be seen as distinct from each other, but very often concerns are overlapping and therefore it is important to keep track of interdependencies among them.
• verifiability means that it is possible to check how requirements are implemented.
• traceability refers to the ability of linking system specifications back to requirements, which can be expressed as goals, problems and opportunities.

The quality of IS specifications is defined by internal consistency and correctness (Snoeck et al., 1999) of all the components and the relationships among them. A
prerequisite to semantic consistency is that the concepts used are unambiguously defined. Semantic integrity can be achieved by identifying and eliminating inconsistencies and incompleteness of system specifications. Inconsistencies between the constructs of diagrams that are represented on different levels of abstractions indicate that two diagrams are incompatible or contradictory. Semantic consistency between graphical representations of requirements on different levels of abstraction is crucial. Completeness refers to whether all the necessary components for the system are included in the diagram. The semantic integrity and consistency of system specifications is essential when engineering artefacts intend to facilitate effective communication of various architectural solutions among business experts and system designers on different levels of abstraction.

1.3. Research Focus

The research focus of my thesis was motivated by the semantic problems of integration in IS analysis and design that are discussed in the previous section. It is necessary to develop a method that provides systematic guiding principles for aligning IS design with respect to the enterprise’s goals.

The aim of a service-oriented modelling method is to provide a set of modelling techniques for modelling early requirements, where pragmatic and semantic aspects of service architectures can be analysed together. It should support a traceable way of bridging pragmatic specifications and conceptual representations of service architectures. Therefore, the main focus of my research is related to three research topics that are important for developing an integrated service-oriented modelling method for IS analysis and design, as listed below.

- **Pragmatic-driven specifications of information systems.** How can a conceptual modelling process be driven by pragmatic considerations?
- **Service-oriented modelling foundation for information systems analysis and design.** How can the concept of service be explicitly used for the analysis and design of information systems and how can the concept of service contribute to the integration of the static and dynamic aspects of IS specifications?
- **Transition principles to implementation-specific design.** How can service-oriented conceptual representations be aligned with implementation-specific design?

Organisational and technical design issues are closely interrelated. IS cannot function in isolation from the enterprise system in which it is embedded (Nuseibeh & Easterbrook, 2000). Pragmatic specifications are supposed to motivate and drive the overall service-oriented analysis and design process (Gustas & Gustiené, 2008). A pragmatic-driven approach is critical as it puts an emphasis on analysing and understanding the ‘whys’ that underlie system requirements, rather than beginning early requirement analyses on detailed specifications of ‘what’ the system should do (Yu, 1997). There is a need for complimentary models (Tolis & Nilsson, 1996) that
show the connection to business goals. At the same time, the pragmatic-driven approach should guide the transition from the ‘why’ models, representing strategic and business-oriented aspects, to the semantic specifications of business processes. It is necessary to develop guidelines that relate the purpose of the system with its structural and behavioural aspects (Kaindl, 1997). A systematic way of modelling beginning with a pragmatic-driven specification and an integrated analysis of static and dynamic aspects of business processes helps to understand and maintain a holistic representation of external and internal views of information system specification (Lankhorst, 2005).

One of the main goals of business managers is to align their business solutions with design, which will be supported by technical components. Business goals motivate the entire development of information systems architecture. Goal modelling (Bubenko, 1993; Bubenko & Kirikova, 1999; Dardenne et al., 1993; Gordijn et al., 2000; Persson & Stirn, 2002; Tolis & Nilsson, 1996; Yu, 1997; Yu & Mylopoulos, 1998) is important as it is used to represent business strategies and decision support (Edirisuriya, 2009). It provides the basis for analysing the requirements, validating the stakeholders’ intentions, defining what designers have to build and verifying the results (Nuseibeh & Easterbrook, 2000). It tells ‘why’ an enterprise has chosen the business rules that govern the entire business of an enterprise (BMM, 2007).

Traditionally, information technology and business processes in organisations are treated separately. Usually, technology experts are not involved in the analysis of strategic decisions and business processes that are actually supposed to move the business forward. The alignment of business with information technology is crucial in achieving the goals of the organisation. Business experts are the ones who constantly update the goals, in order to reflect the changing needs of the organisation, to stay competitive. System experts should also take an active part in the strategic planning and be aware of business decisions and processes. Stakeholders should have access to understandable methods and guidelines for IS analysis and design. To understand why a technical system component is useful, and how it fits into the overall organisational system, it is necessary to develop a method for pragmatic-driven service-oriented analysis, which helps in bridging pragmatic aspects with conceptual models of IS and supports the evolution of business design (Gustas & Gustiené, 2008).

The goal of information systems analysis is to specify the problem domain, without introducing an implementation bias. The creation of computation-neutral models on a high level of abstraction provides business experts with possibilities to understand express and motivate their business solutions. Such models have stronger communicative power, because they are more understandable for people without a technical knowledge. Computation-neutral system modelling could be used both by information systems designers as well as business experts, who play a key role in the semantic integration of information systems and change management of business design. Bridging organisational and technical system parts is crucial for the following reasons:
• it is not technology, but business solutions that should drive the IS development process. Technology is just a means for the implementation of business solutions. Business solutions should prescribe technical components and how they are used for the implementation of business processes.

• service-oriented models presenting the static and dynamic aspects of the IS should be developed according to the goals of a system. Goal modelling should drive the development process, because the change process always starts with a strategy and a business-oriented analysis, where problems and goals are identified.

• functionality that is captured in the business processes should be service-value driven, otherwise services are not useful. An analysis of business processes should be done allowing only value-driven functionality to be introduced (Gordijn et al., 2000; Gordijn et al., 2006). New business solutions can take place only if they are motivated by goals.

A pragmatic-driven method is supposed to provide an integrated way of analysing ‘real-world goals’ (Nuseibeh & Easterbrook, 2000) starting with strategic decisions. Pragmatic specifications could be used as a driving force for a deeper analysis of service architectures. Pragmatic aspects are seldom taken into consideration in the conventional models of information systems analysis and design. They are often analysed on a very high level of abstraction, providing no guidelines to the design. Pragmatic and semantic modelling should be computation-neutral with the purpose of helping to bridge the gap between domain and design experts. Goal models should be established before any specific implementation decisions are taken.

The success of service-oriented analysis and design depends on the fitness between business-oriented analysis and semantic specifications, which define the business process across organisational and technical system boundaries. One of the challenges of service-oriented analysis and design is bridging from pragmatic to semantic IS specifications. Conceptual modelling of requirements in the IS analysis phase involves the use of graphical languages to represent both the static and dynamic aspects of a problem domain. Data description should be integrated with business process description in order to form a comprehensive specification of service architecture. The analysis of the static and dynamic aspects is important since it provides a holistic understanding of the internal and external views of service architecture. The internal view can be defined by using behavioural models. External views can be defined by modelling interaction flows between different enterprise actors that can be viewed as service requesters and service providers.

The dynamic aspects of an enterprise can be captured using the behavioural and interactive models. Both behaviour and interactions are important to understand and fully describe the system dynamics. Internal behaviour is defined in terms of actions and states. Behavioural aspects describe state changes of objects (Gustas & Gustiené, 2009a). Interactions define the behaviour of active objects (Gustas, 2010), which specify why system is useful. Behavioural and interactive aspects are interrelated. They describe different aspects of behaviour, but both are needed to fully define and understand the dynamics of the system. When two enterprise components interact,
one affects the state of the other. In object-oriented approach, state diagrams express state models. Use case, activity and sequence diagrams express interaction models.

**Static aspects** define data, which is an important enterprise asset, which survives different transformations. Data are created, processed and consumed by different organisational and technical components in business processes for different purposes. They define the structural objects in the system, their identity, their relationships to other objects and their properties. The static aspects of enterprise architecture are expressed using semantic dependencies (Gustiené & Gustas, 2008). Models representing structural aspects should capture the entities, attributes and relationships that are important to an application (Blaha & Rumbaugh, 2005). An analysis of the static aspects is important as it provides a basic understanding of active and passive concepts. Active concepts represent different actors of an enterprise (human or technical components) that can be viewed as subsystems. Passive objects represent the internal structure of objects that are characterized by attributes, relationships and state changes.

Various types of semantic models are widely used as a basis to capture the meaning of an application domain as perceived by system designers (Wand et al., 1999). High quality conceptual models should enable early identification and correction of errors (Wand & Weber, 2002). Usually, different models are constructed to define a single aspect of a system. Since all aspects are highly intertwined, it is critical to maintain interdependency relations across multiple diagrams. If the dispersed views and perspectives are defined in isolation, then traceability from one diagram type to another is problematic. The integration of the static and dynamic aspects is necessary to facilitate reasoning and understanding of service compositions across organisational and technical system boundaries. Service architectures are difficult to communicate among stakeholders, if these aspects are analysed in isolation. This is a major reason why this thesis focuses on integration problems of static and dynamic aspects of IS conceptualizations. The benefit of the integration is a holistic understanding of various dimensions of IS specifications in interplay with each other and a separation of concerns.

The third research focus is motivated by the importance of fitness between two ways of modelling: computation-neutral way of modelling and implementation-specific design. This fitness is critical for the success of the final product. This focus is also important for the justification of a new service-oriented modelling method, showing how to bridge to traditional design methods.

1.4. Goal of the Thesis, Research Topics and Questions

The ultimate goal of this thesis is: *The development of a new service-oriented modelling method for information systems analysis and design.*

The purpose of this method is to clarify how different dimensions of service-oriented representations can be integrated on different levels of abstraction. Such a
method is necessary to facilitate reasoning on the quality of architectural solutions across organisational and technical system boundaries. The semantic quality of requirements representations is essential for bridging a communication gap between business managers and IT experts (Nilsson et al., 1999). It facilitates an understanding of necessary changes. This new method should be able to support traceability and consistency of IS architectures along different perspectives and views. It should help to reason about the semantic quality of IS specifications by identifying semantic inconsistencies and ambiguities of system specifications. This method should provide a way to manage the inherent complexity of the detailed organisational and technical representations. It should also help to maintain consistency of IS specifications on the pragmatic, semantic and syntactic levels of abstraction. This method is new because it is based on a new way of thinking and a new modelling approach for IS analysis and design. The philosophy of this method is based on understanding of the concept of service, which is defined by interaction flows between different enterprise system components, which can be viewed as service requesters or service providers. Components can be organisational or technical. Interaction flows are core elements of basic service-oriented constructs for IS analysis and design, which help to separate crosscutting concerns.

The achievement of this goal implies three sub-goals. These sub-goals correspond to three research topics of this thesis. The content of every research topic and the sequence of modelling phases match the modelling process of the method, which is presented in chapters 4 and 5. The sub-goals are the following:

• to develop a pragmatic-driven specification of information systems analysis and design. The development includes principles, techniques, language and modelling notation for pragmatic specifications for IS.

• to develop a service-oriented modelling approach for the integrated analysis of process and data. The development includes a method, principles and techniques of semantic integration of static and dynamic aspects, service-oriented modelling process, notation, and analysis patterns.

• to align service-oriented models with implementation-specific design. This process includes transition principles from service-oriented modelling to implementation-specific design.

All the research studies and results presented in the thesis are related to three research topics, which give the answers to the following research questions:

• How can a conceptual modelling process be driven by pragmatic considerations?

• How can the concept of service be used explicitly for the analysis and design of information systems?

• How can the static and dynamic aspects of IS specifications be integrated at conceptual level?

• How can service-oriented conceptual representations be aligned with implementation-specific design?

The first research question concerns the first sub-goal. The second and the third research questions concern the second sub-goal and the fourth research question concerns the third sub-goal. My research studies and the answers to these questions
helped to develop a new service-oriented modelling method that is the main contribution of this thesis.

The three research topics presented in the thesis concerns the above mentioned sub-goals. The three research topics are listed below:

- Pragmatic-driven specification of information systems,
- Service-oriented modelling foundation for information systems analysis and design,
- Transition principles to implementation-specific design.

The relationship between these topics is based on the service-oriented modelling process. This process takes place on three abstract modelling levels: pragmatic, semantic and syntactic. The modelling process is further explained in chapters 3 and 4. The modelling process starts at the pragmatic level, where IS specifications in terms of goals, problems and opportunities is presented. Gradually, pragmatic specifications are refined into service interactions between service requesters and service providers. Service interaction loops are further analysed at the semantic level, where the static and dynamic aspects of the business processes are integrated using service-oriented constructs. The third research topic was necessary for the alignment of computation-neutral modelling with implementation-specific design. This research topic provides transition principles for bridging a service-oriented way of modelling with the traditional design methods. All three topics together provide a systematic, constructive and consistent way of modelling through three levels of abstraction that must be taken into account in IS modelling (Falkenberg et al., 1996). Theoretical motivation of three research topics with concerning research questions are presented below.

- Pragmatic-Driven Specification of Information Systems

This research topic concerns the first research question: How can a conceptual modelling process be driven by pragmatic considerations?

It is necessary to create a method for structuring the pragmatic knowledge about services, because such knowledge provides motivation for various configurations of service architectures and defines the ‘why’ aspect of the problem domain. Pragmatic knowledge, expressed in terms of pragmatic entities such as goals, problems and opportunities, provides motivation for conceptual representations of enterprise components. Any enterprise business process fragment can be defined as a composition of service conceptualizations. From a pragmatic point of view, a service can be regarded as a problem, opportunity or goal. Pragmatic dependencies should help business experts to analyse the intentions of different actors and contradictions between specifications. Goal hierarchies can help to identify missing processes and data. Goals also provide a basis for reasoning about the semantic incompleteness of system specifications. The refinement of goals in terms of semantic representations is viewed as a driving force in IS modelling and semantic integration process.
Pragmatic specifications aim to provide motivation for conceptual representations of enterprise components at the semantic level that defines business processes across organisational and technical system boundaries. Coherence between pragmatic and semantic levels is important for traceability from business-oriented analysis to service-oriented analysis, which implies an analysis of the static and dynamic aspects of IS specifications. Semantic and pragmatic models define only essential aspects of service architectures that are not influenced by possible technical solutions. Pragmatic-driven specifications are complementary and they are an indispensable part in IS analysis and design phases. Pragmatic specifications are necessary for several reasons. They motivate service events and show the guidelines over how pragmatic aspects are mapped to conceptual representations, which define the semantics of business design, including the structural, behavioural and interactive aspects of business processes.

- **Service-Oriented Modelling Foundation for Information Systems Analysis and Design**

This research topic concerns the following research questions: *How can the concept of service be used explicitly for the analysis and design of information systems and How can the static and dynamic aspects of IS specifications be integrated at the conceptual level?*

Information systems can be structurally visualized as evolving conceptualizations of service architectures. Changes of IS specifications need to be constantly captured, visualized and agreed upon. The alignment of business design with an IT system design is critical in order to make both organisational and technical parts work more effectively.

The traditional methods do not provide sufficient support for developing service architectures, because it is not clear how the concept of service can be explicitly integrated into semantic models. On the other hand, the concept of service is very attractive, because it is quite well understood in different domains and can be applied as a means of decomposition and for separating crosscutting concerns.

The traditional methods for information systems analysis and design are not based on a service-oriented paradigm that facilitates control of business process continuity and semantic integrity. Service loops that are based on the principle of interaction enable continuity control for the business process model and information system design. The identification of discontinuity is important as it helps to find breakdowns in IS specifications. A service-oriented way of modelling can be used for determination of various technical components and for validation with respect to business goals.

The integration of internal and external behaviour, which is encapsulated in a service concept, provides modelling flexibility. It helps to combine intersubjective and objective modelling perspectives, which facilitates understanding and reasoning about service architecture across organisational and technical system boundaries. Objective perspective implies behaviour that defines the states of objects (internal data) (Hull et
Service orientation supports the integration of various dimensions of service architecture into a single modelling notation that facilitates better semantic integrity control between static and dynamic aspects. Traditional modelling methods tend to divide system specifications into separate parts, which causes integrity problems between interactive, behavioural and structural aspects. Integration of these aspects of IS specifications at computation-neutral modelling level is necessary for reaching a holistic understanding of the problem domain, as well as improving the quality of system specifications. Being computation-neutral, service-oriented diagrams are more comprehensible and can be communicated among business experts and IT designers more effectively as compared to various types of implementation-oriented, object-oriented diagrams.

- **Transition Principles to Implementation-Specific Design**

This research topic concerns the fourth research question: *How can service-oriented conceptual representations be aligned with implementation-specific design?*

This research topic is important for bridging a service-oriented modelling method with the existing modelling approaches. It is also important for showing how service-oriented representations that are computation-neutral can be mapped to implementation-specific design. The Object Management Group (OMG) approved Unified Modelling Language (UML) as a standard for object-oriented system development. It is used as the de facto modelling language in many companies. Nevertheless, UML individual diagram types are clear enough, but integrated semantics among models (defining different aspects of the system) are missing. UML diagrams alone are difficult to apply for semantic integrity control and for business logic alignment with system design.

Object-oriented diagrams that represent different projections of IS are difficult to apply for the conceptual modelling of service architectures. Most object-oriented diagrams are suitable for the definition of the internal structure and behaviour of objects. Service-oriented models and object-oriented models are based on a similar set of modelling constructs. There are no fundamental difficulties in bridging service-oriented method to industrial versions of the traditional modelling language such as UML. The advantage of service-oriented modelling is that it integrates the semantics of various UML diagrams into a single type of diagram. Service-oriented diagrams can be used for justification of object-oriented diagrams. The demonstration of how it is possible to bridge object-oriented models and service-oriented models is important for the development of new service-oriented method and tools for IS analysis and design. Using comprehensive graphical descriptions across different perspectives, and along different architectural dimensions, a service-oriented way of modelling provides the possibility to improve semantic integrity and traceability of service architectures.
1.5. Disposition of the Thesis

At the core of this thesis are a set of papers, where the research results were published. The thesis is presented in two parts. Part I contains six chapters that present the cover of the research. Part II contains all publications.

Chapter 1 provides General Observation Over the Thesis. It is an introductory chapter. The aim of this chapter is to present the problem area, the research focus, the goal of the thesis and the research questions. The chapter also presents the three research topics with the theoretical motivation.

Chapter 2 Research Approach and Methods presents the research approach, methods used and philosophical foundation of the research. It also provides a short description of the pre-knowledge and personal background that contributed to the research and presented results.

Chapter 3 Related Work: Concepts and Background presents the important concepts and their background that are most relevant to the research studies. It also presents the concept of service and the characteristic features of service-oriented analysis and design. Desirable characteristics for service-oriented representations also are presented in this chapter.

Chapter 4 Service-Oriented Modelling Method: Modelling Process and Language presents two foundation-stones of the method. The modelling process that takes place through the three levels and the dependencies used are defined in this chapter. The second foundation stone is the basics of service-oriented modelling language. It presents the meta-model, explaining how concepts are interpreted in the service-oriented modelling method. It also presents and explains the notation of static and dynamic dependencies used for service-oriented modelling.

Chapter 5 Application of the Service-Oriented Modelling Method presents the three research topics that cover all the research studies as well as the application of two foundation-stones of the method. It presents the research results that contribute to the systematic modelling process from pragmatic specifications to implementation-specific design. The first research topic is Pragmatic-Driven Specification of Information Systems. This part demonstrates the pragmatic-driven approach for service-oriented analysis and design. The second research topic, Service-Oriented Modelling Foundation for Information Systems Analysis and Design, presents the core elements of the service-oriented method, as well as the basic semantic constructs and the modelling process steps. This chapter also presents five analysis patterns: sequence, synchronisation, iteration, selection and search that were constructed using service-oriented modelling foundation. The third research topic, Transition Principles to Implementation-Specific Design, is important to explain the application of the service-oriented method in combination with the existing modelling methods. It was demonstrated how to map computation-neutral, service-oriented diagram to software-oriented descriptions in terms of object-oriented diagrams.

Chapter 6, Contribution and Conclusions, presents the methodological overview of all research studies presented in different publications. It also includes the summary of every paper, showing how the results of every research study contribute to the main...
goal of the thesis. This chapter presents the main contribution of the thesis, including
the justification of a new service-oriented modelling method, implications for
different target groups, conclusions and future research.

Part II. *Collection of Publications* presents copies of published scientific articles. All
eleven publications presented in thesis are re-printed with the permission from the
publishers.
Chapter 2

Research Approach and Methods

2.1. Research Approach

The research approach of this thesis has a character of design science. Design science is one of the paradigms in Information Systems (IS) research. The design science paradigm is fundamentally a problem solving paradigm. As the realm of IS research is at the confluence of people, organisations, and technology (Davis & Olson, 1985), the design science paradigm seeks to create new and innovative artefacts, which help the analysis, design, implementation, management and use of information systems more effectively and efficiently (Hevner et al., 2004). There are seven guidelines that characterize the principles of design science. Table 1 presents these guidelines and gives the description on how the research in this thesis corresponds to each guideline.

Table 1: Design-science research guidelines and corresponding description

<table>
<thead>
<tr>
<th>Guidelines</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guideline 1: Design as an Artefact</td>
<td>Service-oriented modelling method was developed.</td>
</tr>
<tr>
<td>Guideline 2: Problem Relevance</td>
<td>Application domain of modelling method is integrated into information systems modelling and design.</td>
</tr>
<tr>
<td>Guideline 3: Design Evaluation</td>
<td>The modelling method was applied in one international research project funded by a Japanese industry. A case study was used as an observational evaluation method for that purpose. It was also applied for the extension of Sebi-framework in research collaboration with a Finnish research group from VTT Technical Research Centre of Finland. In this case, architecture analysis was conducted using the analytical evaluation method. A descriptive evaluation method was applied for the development of a detailed process scenario (seven main steps), with the aim to demonstrate the utility and capability of service-oriented constructs. These constructs were utilised to examine the capability of the specific service-oriented modelling language in the creation of five IS analysis and design patterns: sequence, synchronisation, iteration, selection and search.</td>
</tr>
<tr>
<td>Guideline 4: Research Contributions</td>
<td>Service-oriented paradigm was used to solve integration problem of interactive, behavioural and structural dimensions of conceptual models.</td>
</tr>
<tr>
<td>Guideline 5: Research Rigor</td>
<td>The integrated service-oriented modelling construct was defined on the basis of traditional modelling techniques for the modelling of the business process and business data.</td>
</tr>
<tr>
<td>Guideline 6: Design</td>
<td>Service-oriented diagrams were constructed using incremental and</td>
</tr>
</tbody>
</table>
as a Search Process | iterative modelling processes, which consist of seven steps.
---|---
Guideline 7: Communication of Research | The research results were presented at different conferences and project meetings and published in peer-reviewed articles and book chapters.

According to the first guideline, design-science research must produce a viable artefact in the form of a construct, a method, or an instantiation for a specified problem domain. The focus of the research was on two problems: the integration of the static and dynamic aspects of information systems specification, and the development of a pragmatic-driven specification of information systems. The research results presented in this thesis are generalized into a developed service-oriented modelling method for information systems analysis and design.

According to the second guideline, the objective of design-science is to develop technology-based solutions to important and relevant business problems. The relevant application domain of this new method is an integrated modelling of IS conceptualizations.

Within the third guideline, the utility of the purposeful artefact must be demonstrated via evaluation methods. The following evaluation methods were used in the research: a case study as an observational evaluation method was done for that purpose in the international Lyee project (Gustas et al., 2002), for extension of Lyee software engineering methodology. This case study was done in business environment; the method was tested and semantic power of service-oriented modelling language was studied by Travel Agency, which includes buying trips on Internet. The bridging between a service-oriented modelling method and the de facto industry standard language UML was demonstrated in two minor case studies, which were performed in the master course ‘Advances in IS modelling’. Service-oriented modelling method was applied to extend Sebi-framework. This research was carried out in cooperation with a Finnish research group from VTT Technical Research Centre of Finland. Sebi-framework uses semantic technologies to enable information sharing among separate information systems. Architecture analysis, as an analytical evaluation method, was applied. Descriptive evaluation method was applied, when detailed process scenario (seven steps of service-oriented analysis) was developed around the basic modelling artefact to demonstrate its utility and expressive power in incremental service-oriented analysis process. Service-oriented constructs were applied for creation of the main IS analysis patterns. Expressive power of static and dynamic dependencies used in service-oriented construct was evaluated by defining the main workflow patterns such as sequence, synchronisation, iteration, selection and search.

According to the fourth guideline, effective design-science research must provide clear and verifiable contributions in design foundations, or/and design methodologies. The method presented in the thesis is a new method since a service-oriented approach was used for IS analysis and design. It was developed to solve the existing integration problem among different architectural views. It contributes to the integration of interactive, behavioural and structural dimensions of conceptual
models. Being pragmatic driven, the service-oriented modelling method provides with a guideline for overall IS analysis and design process with respect to an enterprise’s goals.

The fifth guideline of design-science requires that the artefact is rigorously defined and formally represented, coherent and internally consistent. The service-oriented modelling method provides the language for service-oriented constructs that are based on three basic events: creation, termination and reclassification. The composition of these three types of basic constructs is important for a conceptualization of the lifecycle of objects in service interaction loops. It integrates various dimensions of service architecture into one unified modelling notation. The aspects of ‘who’, ‘what’, ‘how’, ‘where’, ‘when’ and ‘why’ (Zachman, 1987) are modelled and integrated using one type of diagram and seven process steps (see chapter 5). The interplay between constructs facilitates integration and reasoning of service composition across organisational and technical system boundaries and enables integration of static and dynamic aspects. Various combinations of static and dynamic dependencies used are capable to express the main workflow patterns. This is a unique feature of a service-oriented modelling method. It was demonstrated by providing five analysis patterns, which were presented in this thesis.

The sixth guideline states that the artefact incorporates a search process to find an effective solution to the problem. Service-oriented diagrams are constructed by using incremental and interactive modelling processes. This process is done in seven steps. The incremental method helps with step-by-step modelling, necessary to distinguish and integrate and objective perspectives, which is critical for the integration of static and dynamic aspects of IS specifications. The interactive way of modelling must be repeated for every explicitly specified goal.

The seventh guideline states that the design-research result must be communicated effectively. Research results were presented in different conferences and project meetings and published in peer-reviewed articles and book chapters. Additionally, service-oriented diagrams are computation-neutral artefacts and are oriented towards both IT-system designers and business managers, without technical background. This is done using a three-level framework as well as systematic and integrated modelling process. Existing modelling languages were designed for computation-neutral specification of information systems.

2.2. Research Methods

This thesis contributes with a new knowledge, which is based on the results from theoretical and empirical studies. The research method used in these studies, as a strategy for research design and data collection, has a character of qualitative research. Qualitative research is about exploring issues, understanding and interpreting phenomena and answering questions (Mayers & Avison, 2002). First of all, the development of a new service-oriented modelling method is based on acquiring data
from theoretical studies. Thus, parts of my research have a nature of research method known as a grounded theory (Glasser & Strauss, 1967). It is a research method that seeks to develop a new theory that is systematically grounded in data gathered and analysed. It should be noted that the development of a new method was continuously analysed in interplay with case study results.

The field of IS analysis and design includes many different information modelling methods and visualization techniques that show the connection and relationships between the various architectural domains, such as organisational structure, business processes, information systems, and technical architecture. Therefore the theoretical studies were conducted with the purpose to obtain and increase a better understanding of existing problems and deficiencies concerning methods and approaches in the field of IS analysis and design. These studies were conducted by reading and analysing the publications on information systems analysis and design methods, especially on methods for graphical representations of information systems specifications. The findings from theoretical studies, the comparison of other methods and methodologies in the field of information systems as well as fundamental theoretical assumptions concerning other scientific disciplines (see table 2) helped to challenge a newly created service-oriented modelling method for information systems analysis and design.

The empirical part of the research includes the modelling of service interactions of real websites (e.g. selling trips on Internet) in order to assess the semantic power of service-oriented approach. It also demonstrated how to bridge the service-oriented modelling method to other approaches such as object-oriented approach. Using various examples, the method was tested and the modelling power of service-oriented modelling language was studied. The examples for applying a new method are intended as illustrations of the method and its relevance to the modelling of structural, behavioural and interactive aspects.

The main method of empirical studies (case studies) was based on service-oriented modelling techniques. Modelling is one of the heuristic tools, meaning ‘to find a way’ or ‘to guide’ along the development way in a larger sense (Maier & Rechtin, 2009). Heuristics has much to do with the knowledge and abstraction of experience. Modelling examples helped to validate the semantic power of a service-oriented method. The same examples are tested by using UML as modelling language (Gustas & Gustiené, 2009a; Gustiené & Gustas, 2008). Modelling as a method for research was used in two projects. The first is an international Lyee project, where the Karlstad University enterprise and system architecture research group was responsible for extension of Lyee methodology by using enterprise modelling approach. Lyee is a software methodology the name of which is composed of the last letter of Governmental Methodology for Software Providence = Lyee. This methodology was developed in Japan, at the Institute of Computer-based Software Methodology and Technology. Cartena Corporation and the Institute Computer-based Software Methodology and Technology were joint sponsors of the project, which aimed to extend Lyee methodology worldwide (Poli et al., 2002).
One of the main objectives of extending the Lyee methodology was the adoption of service-oriented representations that were helpful in solving information systems and software development problems. The major focus of my research was to show how the design artefacts of Lyee methodology can be motivated by using service interactions. Service-oriented method was used to demonstrate how various pragmatic, semantic and logical software requirements can be defined and integrated. The project results were presented in the annual report (Gustas et al., 2002) and in one publication (Gustas & Gustiené, 2002).

The next research project was funded and supported by the Centre for HumanIT at Karlstad University. The research focus of this project was on the human values of information technology. It contributed with knowledge showing how to model organisational and technical system parts by using one integrated model. Modelling principles were presented in the HumanIT report (Gustiené, 2003a).

Comparison as a research method was used in theoretical studies. There are two main reasons for comparison of methodologies and modelling approaches (Avison & Fitzgerald, 2006): theoretical and practical.

- The theoretical reason is to better understand the nature of methodologies, by investigating redundant or lacking constructs of modelling languages in order to perform classification and make possible improvements.
- The practical reason is to compare the semantic power of information systems modelling approaches by using some practical examples.

My reasons for using this method are both theoretical and practical ones, as the aim of the research is to contribute with a new knowledge for solving existing problems in conceptual modelling of information systems such as integration of static and dynamic aspects, bridging organisational and technical system specifications, identification and control of undesirable characteristics of system specifications, transition principles from computation-neutral system modelling to implementation-specific design. Object-oriented approach and UML (Booch et al., 1999; OMG, 2009) as well as Structured System Analysis and Design Method (SSAD) (DeMarco, 1979), and pragmatic system modelling approaches (Schoop et al., 2006) were of greatest interest in my theoretical and practical studies. UML is a standard graphical modelling language for system analysis and design. It is used in many companies. This choice is also based on my involvement in teaching assignments of object-oriented modelling and in system analysis and design courses at Karlstad University.

There were two main elements that were of interest in theoretical studies using the method of comparison: the Philosophy of the methodology and the Modelling. These two aspects are foundational basics of the methodology. The element of philosophy is critical in understanding the nature, an ontological foundation of a particular method because it underlines theories and assumptions that shaped the development of a new integrated modelling approach. Modelling is fundamental to the research method. Modelling is used for specification of the conceptualization of a problem domain from someone’s perception. Models are important as they provide a way to capture the essence of design that can be later implemented without losing
semantically critical information. Graphical models are also a means of communication.

The studies provided me with knowledge that helped me understand, which criteria are important in developing a new modelling method. These criteria:

- **Communication support**: The modelling method should support communication and learning among different stakeholders involved in different phases of development, each specification being free of inconsistencies, discontinuities, incompleteness and ambiguities.

- **Pragmatic considerations**: The goals of potential users of a system should be captured, so that the information system is designed to satisfy these users and assist them in meeting their goals and objectives. If a need and motivation to new changes in IS is not identified, then the role of methods is highly questionable.

- **Problem analysis**: The modelling method should provide a suitable means for expressing and documenting the problems of organisation.

- **Validity of design**: The modelling method should provide sufficient details for detecting inconsistencies, incompleteness, ambiguity or redundancy of system specifications.

- **Modelling of technical and organisational system parts**: The methodology should not only address the technical and organisational aspects of a system, but should support the integration of various modelling dimensions and views.

- **Separation of modelling and design**: This separation ensures that the analysis of the existing system and the user requirements are not influenced by design considerations.

The service-oriented modelling method presented in this thesis supports the listed criteria. The method is pragmatic-driven and supports computation-neutral way of modelling. It provides the possibility to clearly separate analysis, which is not influenced by any implementation solutions. At the same time, it provides the transition principles from a service-oriented way of modelling to implementation-specific design. Three-level framework and service-oriented foundation provide the possibility to conduct a systematic and integrated way of modelling, which enables detection of various undesirable characteristics of system specifications. The ontological nature of service concept, which is defined by interactions, supports the communication principle, which is most important to reach understanding.

### 2.3. Philosophical Foundations of Research

The underlying philosophy of the presented service-oriented modelling method is based on Bunge’s ontological principles (Bunge, 1977) and his ontological definition of a system (Bunge, 1979). These principles are fundamental for the motivation of various conceptual modelling constructs (Gustas & Gustiené, 2009b) of enterprise
modelling approach (Gustas & Gustienė, 2002), and development of service-oriented modelling method. The new service-oriented modelling method presented is not a revolutionary, but an evolutionary one. Service-oriented foundation is influenced by various theories and methods from different scientific disciplines. Table 2 presents scientific disciplines that are sources of influence on the research studies.

Table 2: Scientific disciplines, which had an influence for the research study

<table>
<thead>
<tr>
<th>Scientific Disciplines</th>
<th>Theories and Methods</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistics and Language</td>
<td>Semantic roles (Jackson, 1990), (Fillmore, 1968), Conceptual Dependency Theory (Schank, 1975), Theoretical aspects of cognition and communication (Linell, 1994), (Linell, 1989), (Ricoeur, 1976).</td>
<td>1,6,8,11</td>
</tr>
<tr>
<td>Semiotics</td>
<td>The Theory of Signs (Morris, 1938), Pragmatism (Peice, 1877), Organisational Semiotics (Stamper, 1973), (Liu, 2000).</td>
<td>2,4,7</td>
</tr>
<tr>
<td>SOA and Ontological foundations of Service Science</td>
<td>Service-Oriented Architecture (SOA), (Est, 2005), Ontological Foundation of Service Science (Ferrari &amp; Guarino, 2008), (Spohren, 2007).</td>
<td>6,7,8</td>
</tr>
</tbody>
</table>

There are two relevant paradigms concerning the philosophy of the methodologies: science paradigm and system paradigm (Avison & Fitzgerald, 2006). Science paradigm concerns the questions of what is being observed, what kind of questions should be asked, concerning the problem domain: how the questions should be structured, how the experiments are conducted and how the results should be interpreted. The system paradigm is characterized by its concern for the ‘whole picture’ or holistic approach, which strictly emphasizes the interrelationships between the parts of the whole. According to Bunge (1977), it is not enough to do analysis, the process during which certain concepts and prepositions are examined in order to clarify them; it is necessary to do synthesis, the process under which the frameworks
and theories are created. It is synthesis, not analysis that provides an understanding of reality. A new service-oriented modelling method, presented in the thesis, provides not only the modelling techniques for analysis at every level of abstraction, but also the guidelines for transition from the pragmatic to the semantic level. These guidelines ensure consistency between different levels of abstractions, which is necessary to comprehend the modelling artefact as a whole. This way of modelling supports the principle of analysis and synthesis.

Science paradigm and system paradigm are closely related to the concept of system thinking (Checkland, 1981), the discipline for understanding complexity and change (Maani & Cavana, 2000). Any information system has some degree of complexity. The most important issue in information systems development is how to manage this complexity. Complexity is the common denominator of the problems concerning current state of organisations. According to Backlund (Backlund, 2002) complexity is a qualitative concept, it is a measure of the perceived effort required to understand and cope with the system. According to complexity theory (M. Jackson, 1983), it teaches how organisations should manage their relationship with the environment. As every information system is embedded in some organisation and organisations co-evolve with their environment, managing relationships between a system and environment is crucial. Organisations should be prepared to respond and adapt to changes in the environment. As Lundeberg (Lundeberg et al., 1981) states:

"The only way to solve complex problems is to divide them into sub-problems until they become manageable. A requirement for this work is that the solution to the sub-problems gives the solution for the problems as a whole and that the division in sub-systems is coherent" (p. 7).

The notion of subsystem is fundamental in the developed service-oriented method. Decomposition into sub-subsystems is one of the principles that support a component-oriented way of modelling. This way of modelling helps to achieve independency of requirements and to introduce clear principles for separation of concerns. Being pragmatic-driven, service-oriented method gives the possibility to use goal decomposition hierarchies (Gustas & Gustiené, 2008), which describe how the various high-level objectives are going to be achieved. Such goal-oriented approaches (Yu & Mylopoulos, 1998) would allow stakeholders to refine and clarify the requirement through an incremental modelling process.

According to Dietz (2006b), complexity can only be mastered if two conditions are fulfilled. The first condition is to have a comprehensive theory about the things whose complexity one wants to master. The other condition is that it is possible to develop appropriate analysis methods and techniques based on that theory. To manage complexity, it is necessary to have an integrated method and a coherent, comprehensive, consistent and concise conceptual modelling approach. Coherent means that the distinguished aspect of models constitutes a logical and truly integral whole. Comprehensive means that all relevant issues are covered, to make a complete whole. Consistent means that presentations are free from contradictions or irregularities. Concise means that no superfluous matters are contained in it. It is crucial that conceptual models used in the modelling process represent only essential
aspects of problem domain in order not to violate the principles of conceptualization (van Griethuysen, 1982).

Epistemology and ontology are also two important concepts contributing to the philosophy of the methodology. Epistemology or the theory of knowledge is the branch of philosophy concerning the study of nature, methods, limitations, and validity of knowledge and belief. It is about the elements concerned with sources of knowledge, structure of knowledge, and the limits of what we can know (Avison & Fitzgerald, 2006). All models and modelling methods should be based on some philosophical background, in order to motivate why it is better to use one or another artefact. Does it constitute the main principles and assumptions of our existence and reality?

Ontology is one of the important concepts contributing to the philosophy of the methodology. Ontology, according to Oxford Advanced Learner’s dictionary (Hornby, 1974), is the theory concerned with the nature of existence. In philosophy, ontology (from the Greek word of being and science, study, theory) is the study of being or existence. It concerns the nature, the essence of things and the nature of the world these things belong to. It is about understanding of the most general categories needed to construct the description of reality and tell how these categories are related (Milton, 2007). According to Schank (Schank, 1975), conceptualization is a basic unit of meaning out of which thoughts are constructed. Formal representation of knowledge is based on conceptualization: the objects, concepts and other entities that exist in an area of interest and the relationship among them. Ontology could be seen as a specification of conceptualization. Gruber (1995) defines the notion of conceptualization as:

“The objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that holds among them. It is an abstract, a simplified way in which we want to represent the world for some purpose. Every knowledge base is committed to some conceptualization, explicitly or implicitly”. (p. 3) (Guarino et al., 2009).

The notion of ontology as applied to the system helps to better understand the construction and operation of information systems. The ontological perspective of the concept of service allows studying and understanding its nature (Ferrario & Guarino, 2008). It helps to define the elements of service and their relationships. The foundation of service ontology should stipulate how the general similarities and differences of service architectures can be defined. It must be able to address the fundamental semantic issues on how service structures are built and processed. Ontology should not only express the static semantic structure, but it should also be able to define the dynamic aspects of services. Ontological representations of services should be complimented by a capacity to define the actors’ interaction or interoperability dependencies. Enterprise system representations can be built by using semantic descriptions of services. Service requesters and service providers are actors, which must be analysed with other service elements.

The service-oriented method presented in this thesis helps to manage complexity by using an integrated service construct and pragmatic-driven modelling approach. This method defines the modelling foundation (model and notation) of modelling process
and techniques, which are defined on three levels of abstractions. The method is pragmatic-driven, as goals are driving the overall system development process. A starting point of pragmatic-driven modelling method is the analysis of pragmatic aspects such as goals, problems and opportunities. The method also provides the possibility of mapping pragmatic aspects to conceptual representations, which define the semantics of business design. This transition is critical as it is viewed as a driving force of service-oriented analysis and design.

2.4. Personal Background

My main research area is service-oriented modelling, with a focus in pragmatic and semantic aspects of information systems development. It is combined with my pre-knowledge in natural language and communication studies. My interest in semantic aspects of language was one of the reasons for studies of linguistics at Karlstad University. Studies in linguistics (see table 2) on semantic roles of natural language (Gustiené, 1999), was the beginning of the research on the problems of semantic ambiguity. As modelling language plays a major role in information systems development, it was important to investigate how the semantic roles of concepts create structural discrepancies of semantically equivalent representations. Structurally different, but semantically equivalent enterprise architecture fragments create difficulties in communicating system design solutions among stakeholders.

My further studies in the Department of Information Systems were related to a deepening knowledge concerning graphical design languages and their communication support in the context of IS design. The pre-knowledge in linguistics and communication influenced my research platform in the area of information systems analysis and design. It gave me a better understanding of the foundations of natural language, semiotics and ontology. Confluence of knowledge from these areas gave me a better insight into the conceptual modelling languages.

I am a member of enterprise and systems architecture design group at the Department of Information Systems at Karlstad University. This research group is engaged in developing a new graphical modelling approach for information systems analysis and design. It is expected to improve the ability to design enterprise architectures and information systems according to business requirements, and to support communication among stakeholders in information system development. My contribution towards reaching this goal is the development of the pragmatic-driven service-oriented modelling method presented in this thesis. It includes pragmatic-driven specification in terms of goals, problems and opportunities as well as transition principles from a pragmatic level to a semantic level. A case study was done to demonstrate the refinement of pragmatic entities into service architectures. Pragmatic-driven analysis method was developed on the basis of my studies. In service-oriented modelling method, pragmatic entities are analysed as service interactions. Service-oriented modelling foundation contributes with service-oriented
constructs for integration of static and dynamic aspects. One of my most important research contributions is the creation of analysis patterns, by applying service-oriented constructs. I have also defined service-oriented modelling process, notation and modelling techniques. My research studies have also contributed to the modifications of modelling language, which supports integration of behavioural, interactive and structural aspects of service architectures. Although implementation design issues are not in the focus of my research, one of my contributions in this thesis is the presentation of transition principles from service-oriented modelling, which is computation-neutral, to implementation-specific design.

Service-oriented modelling method is a new modelling approach, which is used in our research group. This method also provides a new way of enterprise modelling and integration including component-oriented software system architectures (Jakobsson, 2009) and database view integration (Bellström, 2010). It is intended for graphical design that enables reasoning about system architecture across organisational and technical system boundaries. This method is expected to be used as a basis of future research in the development of a new generation of computer-aided enterprise system engineering tools.
Chapter 3

Related Work: Concepts and Background

3.1. Information Systems Analysis and Design

An information system is a system that collects, stores, processes and transforms information (Langefors, 1973). Every information system works within the context of an organisation and must satisfy its requirements (Bennett et al., 2002). Information systems are developed to support information exchange in organisations for the purpose of improving effectiveness and efficiency of the organisation (Hevner et al., 2004). The role of any information system in an organisation can be seen as a mediator between business and the information technology infrastructure of the organisation (Whiteley, 2004). To determine how a computerized information system can support a particular business activity, it is important to comprehend how the activity is performed and how it contributes to the objectives of the organisation. To develop and understand information systems, it is necessary to have a good understanding about the context of the organisations that use those systems. Information systems development is the way in which information systems are conceived, analysed, designed and implemented (Avison & Fitzgerald, 2006).

Information systems analysis and design are two distinct activities in the process of the information systems development process. These two activities are based on understanding the objectives of the organisation, its structure and processes as well as how information technology (IT) can be used to the business’ advantage (Hoffer et al., 2004). It is necessary that supporting technical system specifications are motivated and justified in the context of organisational process models. System designers need to understand a technical system architecture and organisational infrastructure, where the application is going to be installed. The key issue here is determining the true IT needs and how these needs are integrated into the overall organisational system to support it.

Information systems analysis and design processes involve different stakeholders with different purposes and backgrounds. Analysis focuses on a better understanding of the requirements before attempting a solution. It is the study of the existing system and the identification of the requirements for their solutions (Laudon & Laudon, 2006). The purpose of the analysis phase is to figure out what the business needs (Dennis et al., 2005), to capture the big picture but avoid implementation details (Arlow & Neustadt, 2002). What is important with the analysis phase is the ability to specify the problem domain fully without introducing a bias to any particular implementation (Blaha & Rumbaugh, 2005).

The system analysis phase answers the questions of who will use the system, what the system will do, and where and when the system will be used (Dennis et al., 2005). This
phase concentrates on system requirements determination, the purpose of which is to provide a definition of the functional and other requirements that the stakeholders expect to hold in the implementation (Maciaszek, 2005). It determines the business requirements for the new system. It specifies what must be done, not how it should be done. During the analysis process, developers consider available sources of information and try to resolve possible ambiguities. Analysis is the process of creating models that capture the essential characteristics of the desired system.

Design is a way of checking that the planned solution really meets the needs of the situation, before it will be put into practice (Bennett et al., 2002). System design provides the specifications for the information system solution defined at the analysis phase. Design is sometimes viewed as a progressive refinement from abstract to concrete and specific (Maier & Rechtin, 2009). It specifies how the system will fulfil the objectives defined at the analysis phase. It shows how technical and organisational components fit together (Laudon & Laudon, 2006). The process of design is difficult, because it is not a purely analytical task. Design requires synthesis. It is a process, where it is necessary to find a way to bridge the gap between the desired features of the system with available resources (Blaha & Rumbaugh, 2005); it is the task of how to fit the entire system together clearly.

To deal with this complexity of information systems and to reach a successful system development process, much depends on good defined descriptions of enterprise architecture. This architecture provides a foundation for organizing different aspects and components of any information system that is supposed to be developed. Design and architecture of the system embody the most important decisions about how the system is build and implemented.

### 3.2. Enterprise Architecture

To control or manage the necessary changes and to understand the enterprise system as a whole, the organisation has to have an architecture that guides the design and evolution of the enterprise. Architecture expresses the characteristic features, structure, behaviour and interaction relationships of a specific artefact (McGovern et al., 2004). The Wikipedia encyclopaedia (Wikipedia: The Free Encyclopedia, 2007) gives the following definition of the concept architecture:

“Architecture is the art and science of designing buildings and structures. It is a subjective mapping from a human perspective to the elements or components of some kind of structure or system, which preserves the relationships among the elements or components”.

According to the IEEE Standard (2000), architecture is the fundamental organisation of a system consisting of components, their relationships to each other and to the environment, and the principle guiding its design and evolution. The architecting process is important in bridging the gap between strategic issues of the organisations and the implementation of them. As every building or construction needs the architecture the same needs the enterprise in its evolitional process.
Architecture in the context of an organisation is referred to as 'enterprise architecture'. The concept of enterprise in the context of information system development denotes a limited area of activity in an organisation (Bubenko, 1993) that is of interest to different stakeholders. According to Ed (2003), an enterprise is an ever-evolving entity constantly changing in response to external and internal influences. It is formed to produce a product or provide a service (O'Rourke et al., 2003). It is a purposeful system that is supposed to create value, which is typically expressed by vision, goals or achievable objectives (Marshall, 2000). Enterprise is a broader understanding of an organisation. It is a complex and holistic system, encompassing organisational, business processes and technical views (Nightingale & Rhodes, 2004), which need to be architected. In this thesis the concept of an enterprise system is understood as any problem domain: it could be an organisation, or a set of organisations that supports their businesses with computerized information systems.

Enterprise Architecture (EA) is the collection of strategic and architectural disciplines that embrace information, business and technical architectures (Perks & Beveridge, 2003). It is an approach for aligning business and IT issues in a company (Langenberg & Wegmann, 2004). Enterprise architecture is important as an instrument to address the enterprise-wide integration of different architectural domains (Lankhorst et al., 2004b). EA is important in the context of information systems development as it provides insight into the organisation's structure, business processes and technology that support the business and make up the whole enterprise (Steen et al., 2005). It describes a coherent whole of the principles, methods and models that are used during the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure (O'Rourke et al., 2003). Such architecture provides possibilities to understand and determine the necessary needs for changes, integration and alignment of business and technology issues (Lankhorst, 2005). It provides a basis for systematic analysis of organisational and technical services. It is especially important for stakeholders to feel that they are the part of the project or knowing where organisational and technical components can be found (Hoppenbrouwers et al., 2005).

There are a number of valuable concepts that are significant and reflect the general architectural description of an enterprise. These concepts are necessary to comprehend the information system development. The key concepts and relationships among them, which are related to Enterprise Architecture (EA), are presented in figure 1. This figure is a modified version of EA concepts and relationships presented by (Lankhorst, 2005).
An information system is embedded in an enterprise system. Autonomous services are motivated by different enterprise system goals. Architecture defines the enterprise system, which is expressed by the views of different stakeholders and can be represented in different dimensions. Behind every graphical representation lies the description of the stakeholders' view that together provide the integrated architectural description of the enterprise system. Dimensions are based on different models that are defined by various modelling languages. Semantic models, which are viewed as meta-models, define the modelling notation of graphical language. The modelling language is necessary for an unambiguous understanding of diagrams and for communication among stakeholders. The purpose of communication is to bridge the gap using shared models and methods with the purpose of developing the comprehensible descriptions of EA.

The design of EA provides the ways to deliver business solutions that are enabled by technology. EA is characterized by the use of frameworks, which support the analysis from business-oriented level down to the IT level. The framework can be seen as a key to understanding enterprise architecture, the dynamics of the organisation. The Zachman framework (Zachman, 1996) is one of the frameworks that comprise the enterprise architecture. It is a two-dimensional schema representing six perspectives, as viewed by the planner, owner, designer, builder, subcontractor and functioning system. To understand the significance of the creation and construction of a new artefact, it is necessary to examine the future artefact creation from six different aspects (O'Rourke et al., 2003): motivation, organisation, and location, materials, building process and timing coordination. The framework represents six columns for defining different aspects: things, processes, connectivity, people, timing and motivation. Typically, it is defined in various perspectives such as

**Figure 1:** Concepts concerning enterprise architecture

Table 3: Matrix of various representations of enterprise architecture

<table>
<thead>
<tr>
<th>Aspects/Views</th>
<th>Data (What)</th>
<th>Function (How)</th>
<th>Network (Where)</th>
<th>People (Who)</th>
<th>Time (When)</th>
<th>Motivation (Why)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope Planner view</td>
<td>List of concepts</td>
<td>List of processes</td>
<td>List of locations</td>
<td>List of organisational units</td>
<td>List of business events</td>
<td>List of business goals</td>
</tr>
<tr>
<td>Organisational system Owner view</td>
<td>Entity Relationship diagram</td>
<td>Business Process diagram</td>
<td>Diagram of logical network</td>
<td>Organisational Decomposition chart with roles</td>
<td>Schedule charts</td>
<td>Business Strategy Plan</td>
</tr>
<tr>
<td>Information System Designer view</td>
<td>Logical Data Architecture</td>
<td>Software Application Architecture</td>
<td>Distributed system Architecture</td>
<td>Human Interface Architecture</td>
<td>Control structure</td>
<td>Constraints and Rules</td>
</tr>
<tr>
<td>Technology Builder view</td>
<td>Physical Data Architecture</td>
<td>Deployment Architecture</td>
<td>Technology Architecture</td>
<td>Presentation/Layouts structure</td>
<td>Component control structure</td>
<td>Rule design</td>
</tr>
<tr>
<td>Representations Subcontractor view</td>
<td>Data definition</td>
<td>Process design</td>
<td>Network Architecture</td>
<td>Interface Architecture</td>
<td>Timing definition</td>
<td>Rule specification</td>
</tr>
<tr>
<td>Functioning System</td>
<td>Data</td>
<td>Components</td>
<td>Network</td>
<td>Organisation</td>
<td>Schedule</td>
<td>Strategy</td>
</tr>
</tbody>
</table>

Enterprise architecture refers to various types of graphical representations that define different architectural domains. It defines how business, data, technology and software application structures (Spewak, 1992) are perceived from different points of view. The ‘what’ represents structural aspects, the ‘how and when’ represent behavioural aspects, and the ‘where’ and ‘who’ represent interactive aspects of enterprise architecture. According to service-oriented framework (see figure 3), the ‘why’ dimension corresponds to the pragmatic level. At this level, business-oriented analysis is done. The analysis results in pragmatic specifications, which motivate and drive further analysis and design processes at the semantic and syntactic levels. Structural, interactive and behavioural aspects are integrated by using service-oriented modelling method at the semantic level. This is a conceptual level, which corresponds to Owner view in Zachman’s framework. The logical design is represented by the syntactic level, where implementation-oriented projections of various diagrams can be populated. There are different ways for organisations to create value, but value creation is always done in work process, which can be seen from different perspectives. Integration of these perspectives will contribute to understanding and possibility for creating and enhancing business value to the enterprise (Rouse & Baba, 2006).

Another view of enterprise architecture is presented by Lankhorst et al., 2005. It consists of organisational architecture representing organisational design, including organisational actors, their roles and relationships in the organisation. Information
architecture represents necessary data and its structure. Process architecture is important for representing business processes. Application architecture represents the application logic. It presents different application components and how they relate to each other. Technical architecture represents the architecture of the technical infrastructure, needed to run applications. Figure 2 presents different domains necessary to comprehend enterprise architecture as a whole.

Figure 2: Heterogeneous architectural domains
(Adaptation of Lankhorst et al, 2005, p. 50)

Semantic integrity among different architectural domains is critical for understanding and communicating clearly the enterprise architecture. To get an integrated view of an enterprise, it is necessary to integrate various dimensions for describing architectures of an enterprise in a holistic view. Integration of heterogeneous architectural domains is crucial, as integration eliminates redundancy, discontinuity and differences within the enterprise (O’Rourke et al., 2003). Achieving integration is problematic due to differences in architectural modelling methods and models (Steen et al., 2005) and the lack of a common language for shared knowledge (Falkenberg et al., 1996). Different methods and models from different architectural domains make it difficult to integrate personal views into single enterprise architecture. Another important issue is how structural and behavioural aspects are related to each other (Lankhorst, 2005) as well as how they are linked with organisational and technical domains.

The quality of system specifications depends on reaching a semantic integrity of graphical representations. Semantic integrity is the degree to which graphical representations are comprehensible despite their complexity. Semantic integrity issues in conceptual representations create difficulties in communicating semantic details of
the application domain. Ambiguous, incomplete and inconsistent IS specifications make the verification of semantic integrity between business process and data especially problematic. The semantic relations used in conceptual modelling are one of the key constructs to capture how concepts are related to each other. Semantic integrity of IS specifications could be reached by eliminating inconsistencies, incompleteness, ambiguities and redundancies during modelling process. Semantic integrity of a diagram type can be checked using semantic relations between elements of the models, which is critical for achieving of right system design.

Usually various models or a model collection are used to represent these aspects. The problem is how to reach consistency between models in order to have a complete and integrated enterprise system specification. The lack of conceptual modelling approaches which can support verification of semantic integrity among different graphical representations is one of the biggest problems for information system designers. Inconsistent, incomplete and ambiguous conceptual representations of EA create difficulties in verification and validation of technical and business solutions. It results in an inability to bridge a communication gap among business experts and system designers.

3.3. Enterprise Modelling

Enterprise engineering is a branch of requirements engineering (Bubenko & Kirikova, 1999) that deals with modelling and integration of organisational and technical knowledge about business processes and business data (Gustas & Gustiené, 2004). There are two major challenges facing the overall enterprise engineering process: enterprise modelling and integration (Vernadat, 1996). One of EM goals is to align organisational processes with technical system operations. It helps system developers to visualize, specify, construct and document the interactive, structural and behavioural aspects of an enterprise system. EM is the process used to analyse the current situation, and to design future solutions to identified problems (Jörgensen, 2009). It can be seen as one of the aids for improving the communication among different stakeholders by providing them with relevant methods, techniques, notations and models. It includes modelling interplay of the computation-neutral business domain and the technology related implementation-specific aspects. It is a process that helps to understand business and to make improvements of its performance by using different enterprise models, which can be seen as the systems of knowledge about the enterprise (Bubenko, 1993). These models are used for the purpose to provide a conceptual representation of enterprise architecture including entities, relationships, processes, information resources, actors and their relationships. According to Stirna and Persson (2009), there are two main reasons for applying EM: for developing business and for ensuring the quality of the business. Enterprise models are used for communication purposes during the development process. They help to analyse and redesign business processes.
Complexity and different perspectives on IS makes multiple diagrams inevitable. It should be noted that different views and perspectives do not make integrated enterprise architecture. To obtain value from the graphical representations that are used in an organisation by different groups of people, these documents must be integrated. Integrated enterprise models might help business and IT experts to define, visualise, assess and trace the impact of organisational changes from one view to another. Therefore, they should be considered as a corporate resource in diagnosing potential problems. Enterprise models are also crucial to enable reasoning about business process integrity and the purposeful implications of an organisational change. The outcome of the early enterprise modelling phase can be defined in terms of various types of pragmatic specifications, which are referred to as goal models (Mylopoulos et al., 1999). The later EM phase is important for justification of implementation-specific design, including the modelling of software components, their interactions in terms of interfaces with the structural definition of the printout, message, screen and file layouts. These layouts can be understood as syntactic elements of implementation-oriented diagrams (Gustas & Gustiené, 2002; Gustas & Jakobsson, 2004).

Integrated models used during the middle enterprise modelling phase can be a useful tool for the users and the system developers to check consistency, since consistency rules are an integral part of the model (Glinz, 2000). The creation of integrated models that have no implementation bias, provide business experts with possibilities to motivate their business solutions. Such models have stronger communicative power and are more understandable for people without technical background. Business solutions presented by integrated conceptual modelling constructs provide better guidance for the analysis of implementation related specifications. They are crucial for linkage between computation-independent and implementation-specific representations of IS. Computation-independent specifications being implementation agnostic, and built on an integrated set of semantic and pragmatic dependencies, can be used as a tool for bridging the communication gap between business and system development experts. Interplay of various EM representations that can be presented on different levels of abstraction is discussed in the next section.

3.4. Model Driven Architecture and Modelling Levels

Model Driven Architecture (MDA) (OMD, 2010) is an approach for model-driven engineering of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as three models: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). MDA aims to provide an open, vendor-neutral approach to the challenge of business and technology change. The purpose of this approach is to separate the
functional specifications of a system from implementation issues on a specific platform (Grangel et al., 2008). This framework comprises three abstraction levels and mappings between them. Platform independent models are built using UML and other associated OMG modelling standards. CIM covers the business aspects of an enterprise system. This model is sometimes called domain or business model, which describes the situation in which IS will be used. No information about automated data processing system is covered at this level. By analysing and modelling enterprise systems and not concentrating on the specification of desired functionality, system developers are able to better manage the complexity of developed information systems. Separation between enterprise models and implementation-specific models leads to a natural division of IS engineering products in two different representations. This division is important, because it is more reasonable to conceptualize enterprise architecture before the supporting software system is defined (Checkland, 1981).

PIM describes the operation of a system, but hides the details of a specific platform. PSM combines the specifications from the PIM level and provides specification on how the system uses a particular type of platform. The important feature of MDA is the mapping rules and techniques used to modify one model into another model. Still, a question as to extent the automation of mapping rules between CIM and PIM levels is feasible remains a major research effort (Lankhorst, 2005). To understand how and why technical system components are useful and how they fit into the overall organisational system, at least three modelling levels of information system specifications are necessary; pragmatic level, semantic level and syntactic level (Falkenberg et al., 1996). Interplay between MDA and three modelling levels are graphically presented in figure 3. This figure explains similarities between MDA and IS modelling levels.

![Figure 3: MDA framework and three modelling levels](image)

According to FRISCO report (Falkenberg et al., 1996) pragmatic, semantic and
syntactic levels are of great interest in the context of information systems design, as they deal with the usage, meanings, and structures of representations. These three levels can be viewed as three dimensions of requirements engineering: agreement, representation and specification (Pohl, 1993). The agreement dimension concerns the pragmatic aspects of change analysis. The representation dimension deals with semantic aspects of the system, representing static and dynamic aspects of business processes across organisational and technical system boundaries. The specification dimension can be compared with syntactic level where implementation-oriented details are defined that explains the data processing needs of a specific application or software component (Davis & Olson, 1985; Gustas & Gustiené, 2004). The pragmatic and semantic levels can also be compared with three dimensions presented by Nellborn and Nilsson (1999) for business modelling: intentional, concept and organisational. The pragmatic level can be compared with intentional dimension, which describes the strategic dimension in terms of goals, critical success factors, strengths and weaknesses. It gives the answers to ‘why’ questions concerning business. The semantic level can be compared with concept dimension and organisational dimension. Concept dimension describes business entities of domain and their relationships. It concerns the question ‘what’. Organisational dimension gives answer to the question ‘how’: it describes actors, activities and workflows.

Three-level framework, which is presented in figure 3, can be used to understand various dimensions of IS system analysis and design. A clear modelling method of how to bridge these three levels is crucial for the development of a comprehensible and systematic modelling process. Models used at every level should contribute to the overall methodology of enterprise-wide engineering (Gustas & Gustiené, 2004). Such framework provides not only a way to structure business and technological solutions from a top-down perspective, but also provides a way in which the bottom-up development of services can match top-down business requirements.

Model-driven development should be focused on pragmatic and semantic issues of IS. The pragmatic and semantic levels concern computation-independent modelling. At these levels, information systems are analysed as they are perceived by business experts and users, neglecting how the system will be realized (Perrone et al., 2005a). The syntactic level should be mainly focused on computation-specific, but platform independent issues, which are necessary to understand how pragmatic and semantic models will be implemented.

A starting point of information systems engineering is to deal with visions, principles, goals and objectives that characterise IS architecture. Since this activity concentrates on business-oriented analysis concerning the strategic description and transitional processes, it will be referred as pragmatic-driven information system engineering. The pragmatic level is the level where business-oriented analysis should be done in terms of pragmatic entities as goals, problems and opportunities. Goals provide motivation behind new business solutions. The analysis of goals helps estimate the completeness of EA and to identify business processes without any motivation. Goals justify the presence of organisational and technical components.
They explain responsibilities of different enterprise actors as well as provide a basis for conflict detection and resolution (Gustas & Gustienė, 2008).

Goal-modeling techniques have long been recognized and accepted as an important and effective aid to requirements engineering processes. Goals are seen to have power in supporting the elicitation and elaboration of requirements (van Lamsweerde, 2004; Yu & Mylopoulos, 1998). The notion of goal was used in requirements engineering process by many researchers for many reasons. For example, KAOS goal-directed language and method (Dardenne et al., 1993) is used for requirements acquisition. Goal-modeling techniques have also been used in modeling the relationship between the intended system and the environment in which it will operate (Habli et al., 2007). This relationship can be expressed in goal-based dependencies. For example, the approach (Mylopoulos et al., 1999; Yu, 1997) concentrates on modeling the intentional relationship between actors of enterprise system. In this approach, it stands for distributed intentionality, stating that actors are related to each for reaching their goals. In Business Motivation Model (BMM) (OMG, 2010), the basic notion is that of goal, which emphasises what a business seeks to reach, a desired condition or a state of affairs. This model includes the notion of means as an instrument to achieve a goal. Goal-modeling technique here aims to determine what various actors want to achieve and how (Gordijn et al., 2006). These techniques are also used to analyze conflicts. Reaching one goal can cause problematic situations for reaching other goals. Goals have been used in scenario-based approaches (Potts et al., 1994), as well as for dealing with non-functional requirements (Chung & Leite, 2009). It should be noted that designers have problems in dealing with the fuzzy concept of ‘goal’ in spite of the fact they need to find out the goals of real systems. Goal-modelling should be supported by the necessary methods and techniques to be meaningfully performed (Rolland et al., 1998).

After stating goals and analyzing all the possibilities stated at pragmatic level, it is important to show how these goals will be defined by using conceptual models at the semantic level. Pragmatic specifications, justifying the question ‘why’ should drive and guide the overall IS engineering process. The semantic level is important for the integration of interactive ‘where’ and ‘who’, structural ‘what’ and behavioural ‘how’ and ‘when’ aspects (Zachman, 1987) of conceptual representations. This level should provide the possibility to identify undesirable characteristics of system specifications such as inconsistency, incompleteness, redundancy, ambiguity and incoherence. Semantic descriptions constrain the development of implementation-specific models at the syntactic level. The syntactic level is necessary for technology-oriented analysis. This level defines the details, which explain the data processing needs of a specific application or software component.

Considering the entire information systems development process, the presented three levels cover both system analysis and the design phase. The double-headed arrows express the need for fitness and alignment between levels. All three levels are interrelated, as they provide a foundation for understanding the enterprise system as a whole. Such three-level architectural framework is the foundation of modelling that helps to provide interplay between business needs and technical solutions. The
pragmatic level prescribes and motivates information systems specifications at the semantic level. Information systems specifications cannot be understood unless the goals are stated. The analysis done at the semantic level must fit with the stakeholders’ goals stated at the pragmatic level. Coherence between these levels is crucial in reaching consistency among IS specifications.

3.5. Conceptual Models and Modelling Language

Conceptual modelling is a fundamental activity in requirements engineering (Nuseibeh & Easterbrook, 2000). It is the act of abstracting a model from a problem domain (Lankhorst, 2005). Modelling is not an objective in itself, but it is a means to communicate, analyse and design the objects of a problem domain (Biemans et al., 2001). There are three terms that are important in setting up a modelling framework: models, language and method. The models constitute the fundamental basis for information upon which business experts and system developers interact (Pons & Baum, 2000). It is a purposely abstracted, precise and unambiguous conception (Falkenberg et al., 1996), the denotation of which is a representation of a model in some language. As all modelling artefacts have to be communicated, the modelling language plays an important role in the modelling process. Modelling language is an essential instrument for the presentation of different solutions and the communication of service architectures. Modelling language should have a strong visual power as per the popular saying that ‘a picture is worth a thousand words’. Its purpose is to specify, visualize, construct and document the artefacts of the system.

Conceptual models are created to represent the abstraction of desired reality, as perceived by someone. They are intended to capture knowledge about the real-world domain (Wand et al., 1999) and represent complete and consistent specification of the system to be built (Kruchten, 2006). According to the Institute of Electrical and Electronics Engineering (IEEE) standards, a model is an approximation or a representation of selected aspects of the structure, behaviour, or other characteristics of a real-world process or system, and help to control information system specifications, its design and its production plan (Maier & Rechtin, 2009). Models of IS and subsystems are basic units of IS development and they specify a part of the function, structure and behaviour of the system and are used to:

- Visualize the system as ‘it is’ or as designers want it ‘to be’,
- Specify the structure and behaviour of a system,
- Provide templates to guide us when constructing a system,
- Document the solutions that service developers have made.

It is agreed today that graphical representation of IS specifications is better than the requirements presented by a natural language text. Most people find such representations easier to work with than purely textual representations. If the system specification is expressed using natural language, different people may understand the
same specification in different ways. Because of the ambiguity of natural language it may lead to misunderstandings in design and implementations. Additionally, certain semi-formal graphical languages offer the possibility of reasoning support during later IS development stages. Graphical representations used to build models of the systems are diagrams. A diagram is a graphical representation of a set of elements, which are used while building models. Diagrams are used for communication purposes, to test ideas and make assumptions. The difference between the model and the diagram is that a single diagram documents some aspects of a system, whereas a model provides a complete view of a system at a particular stage and from a particular perspective (Bennett et al., 2002). Models and meta-models (Martin & Odell, 1998) are based on notations that define syntactic and semantic elements for creating language for modelling process.

Modelling language is used to denote the models. It consists of a set of basic concepts and rules determining the possible models that can be denotable using the language. The modelling language has a vocabulary and a grammar. Vocabulary represents elements that are the parts of the model and grammar defines how these elements should be linked to provide the meaning. Inability to reveal and convey the necessary meaning causes semantic problems of communication. Using semantic rules and modelling principles, these elements are used to create graphical constructs that are able to express semantics of the system. As models are a means of communication during the information systems development process, it is necessary to build them taking into account the three levels of semiotic rules: pragmatic, semantic and syntactic (Krogstie et al., 2006; Krogstie & Solvberg, 2003; Morris, 1938; Nöth, 1990). These rules together contribute to the success of the communication, as they help to represent the meaningful or semantically right expression of the communicated message.

The main objective of enterprise modelling as well as business modelling is to bridge the communication and knowledge gap between stakeholders. The method here plays a very important role, as it provides concrete guidelines for a systematic and integrated way of working during the information systems development (Nilsson, 1995). According to Nilsson (1996), three main constituents of a method could be distinguished:

- **Perspectives**: basic principles, views and assumptions which influence the work during the development process,
- **Work tasks**: guidelines to manage different issues and decisions during the development process,
- **Interest groups**: the list of actors and their responsibilities for the work during the development process.

The method should help to bridge the communication gap between business and system people by using shared knowledge in order to reach a successful management of business issues and its links to supporting information systems. Shared knowledge (Falkenberg et al., 1996) is the knowledge that communicating actors assume to be identical to that of others, as resulting from the negotiations implicit in the
communication process. If communication has resulted in shared knowledge, it may be said that actors achieved ‘inter-subjectivity’, which is important to avoid misunderstanding. The method should always be goal-oriented and should guide how to carry out the design following manageable and comprehensible steps, advising which decision to take with respect to the design decisions and goals to reach (Biemans et al., 2001).

3.6. Method Development

The notion of method development has two meanings. It could be understood as an act of being developed into something new, or it could be seen as a growing process. The development of a method is always related to needs and purposes. Either it is necessary to develop a new method, with an aim towards a paradigm shift, introducing new ideas and ways of thinking, or to re-develop existing methods. The development of the method should always be based on the philosophical foundations that clarify how and why the method was developed, and why it is better than already existing methods.

Methods are needed to enable rapid enterprise and system architecture changes. They are necessary to support organisational transformation into agile enterprises that can react quickly to environmental changes and be more competitive. Methods are also necessary for the integration of business processes and business data. A good method should support consistency of data, process, and business strategy throughout an organisation (Finkelstein, 2006). It should also support alignment of business and IT systems that is necessary for effective collaboration among stakeholders.

The development of a method, concerning foundations for system development, is important for the system development life cycle (SDLC). The information systems development life cycle is a traditional methodology used to develop, maintain and change information systems (Hoffer et al., 2004). This cycle features several phases that designate the development process. The method is important for providing the steps and guidelines for the work process at every phase of the SDLC. It should help to record accordingly the requirements in order to make new information systems meet the organisational objectives. Such method supports possibilities for necessary modifications or early changes. The method should help to comprehend various dimensions of enterprise architecture in order to understand why it was developed and what the problems are. It should clearly emphasise the needs for the new design. The method should support views and dimensions of the Zachman Framework (1987), to provide an understanding of enterprise architecture as a whole. It should support computation-neutral design that focuses on the business issues as well as support on technology specific design that focuses on component design (e.g. software application or database file).

The method presented in this thesis is developed for the early and middle stages of
system development life cycle, which corresponds to the analysis and design phases. The early stages of analysis are supported by pragmatic dependencies (see fig. 6) at the pragmatic level, which is important for representation of the ‘why’ dimension. The service-oriented modelling method can be applied for system analysis and design phases, which correspond to owner perspective and supports ‘what’, ‘how’, ‘where’, ‘who’, and ‘when’ dimensions. Being pragmatic-driven, the method supports a systematic way of modelling, starting from business-oriented analysis to service-oriented analysis, which helps to integrate static and dynamic aspects of IS conceptualisations. The method also provides the transition principles to implementation-specific design, which corresponds to designer perspective. The method is based on new modelling principles, modelling languages, processes and analysis patterns. This method helps to unambiguously decompose the enterprise system into subsystems and to separate crosscutting concerns between subsystems. These issues are fundamental for the management of EM complexity.

3.7. Conceptual Modelling in Information Systems Development

The process of conceptual modelling has always been an essential part of IS development (Bubenko, 2005). The purpose of conceptual modelling, in the field of information systems, is to provide a set of constructs, rules and principles that show how to create a representation of a problem domain (Wand et al., 1999). These models are used for the following purposes (Kung & Solvberg, 1986):

- to increase understanding of the problem domain,
- to help the analysis process and serve as the specification of system requirements and a common basis for system design,
- to facilitate better communication, agreement and the learning process between the stakeholders involved during the development process.

Conceptual modelling provides a possibility to identify, analyse and describe the essential concepts and constraints, using modelling language, based on a set of the concepts to form a meta-model (Guizzardi et al., 2002; Nilsson, 1995). At the requirements analysis phase, conceptual models are often used to represent structural and functional aspects necessary for information processing. It also facilitates the detection and correction of system development errors (Wand & Weber, 2002) and highlights omissions and possible inconsistencies that should be resolved. The process of conceptual modelling involves different stakeholders with different views and different assumptions about the embedding environment and tasks. Conceptual models used during this process should be concise and express reality perceived by system users and system designers. They should be understandable for all stakeholders involved. Poor understanding of problem domain is a primary cause of project failure (Yu, 1997). The success of understanding depends greatly on consensus reached among all the stakeholders involved. To make information
exchange possible among different actors, it must be represented in communicable forms (van Griethuysen, 1982). Since communication can be defined as a process of transferring and sharing knowledge in the form of information flows from a source to a receiver as efficiently and effectively as possible, there are two main points in conceptual modelling process:

- The ability of system analysts and designers to represent relevant system requirements (source of information). Conceptual models should represent precisely the requirements agreed upon between system users and system designers,
- The ability of all stakeholders to comprehend unambiguously the presented system requirements (receiver of information). Conceptual models presented by system designers should be semantically clear enough to be understandable for all stakeholders.

Problems in conceptual modelling arise when the nature and the underlying meaning of the elements used in constructing models are unclear. Very often the elements are ambiguous in their usage and meaning. The construction of models should always be motivated and based on philosophical principles that are related to real-world phenomena. The constructs should have a strict semantic and pragmatic foundation. By this is meant the rules of how different elements are combined. How and why elements are related to each other depends on the semantics of the model, which should have enough semantic power to represent both static and dynamic phenomena of problem domain (Wand & Weber, 2002). The ability of the models to capture and represent the necessary knowledge about the problem domain is a prerequisite for the modelling process, as the models should support communication between system analysts, developers and users (Aguirre-Urreta & Marakas, 2008). It is necessary that the meta-models used to form the language for conceptual modelling should have a theoretical foundation that provides sufficient expressive power. Such models can help to sharpen the design and improve the quality of the dialogue between the stakeholders to avoid misunderstanding (Gustiené & Carlsson, 2010) and facilitate learning process.

To understand the graphical representations of the system and how the system works, all necessary architectural aspects should be integrated. Different aspects should be synchronized to ensure consistency and completeness of the overall system specification (Whitten & Bentley, 1998). As different models define different aspects and views of the same system, these views should be interrelated. The integration of multiple diagrams used to represent different models is crucial for successful reasoning, motivation and solving the problem of information systems analysis and design. Integration depends critically on a clear definition of the meta-model and modelling language (Finkelstein, 2004).

Longman Lexicon of Contemporary English (McArthur, 1981) defines integration as: “To make complete from a number of parts.” Integration is a process of merging different subsystems into one system. It involves joining the subsystems together providing the necessary ‘glue’ for a system to function as a whole. To succeed with integration, it is vital not just to know what parts to merge, but to have a method that
supports the integration process. It is important that the language and methods used maintain their relevance throughout the modelling process (Maier & Rechtin, 2009).

Conceptual modelling can be seen as the key to a successful integration process as it allows for a focus on semantics and design of integration as opposed to implementation issues. It is the only way to visualize a system design and validate it against the requirements. Model integration is important since it provides the possibility to combine the static and dynamic aspects of IS specifications. In this thesis, the service-oriented modelling approach enables the analysis of intersubjective and objective perspectives of service architectures using one modelling construct, which helps to integrate the static and dynamic aspects of conceptualizations. This construct is based on modelling interactions between service requester and service provider. It facilitates understanding and reasoning about service architectures across organisational and technical system boundaries. The presented service-oriented modelling method also provides a way of bridging enterprise architectural perspectives, which are necessary to have a holistic view of the system specification and to support communication and learning. It provides the techniques for service-oriented analysis and design starting with pragmatic-driven system specification and bridging it to the semantic descriptions of IS. The systematic way of modelling improves semantic integrity and traceability between representations on various levels of abstraction. Such modelling method makes the IS development process more holistic and seamless (Nellborn, 1999).

3.8. Concept of Service

The notion of service is not new and the meaning of it is intuitively understandable for people with different backgrounds. Recently, this concept has been shaped and used in different ways. Even if the concept is used in different areas, there is no properly identified definition of the notion of service with what people are commonly referring to when asking for a service. One of the reasons is that relatively little attention has been paid to studying and understanding the real nature of the notion of service concept. It is necessary to have a clear understanding of what a service is for its evolution and redesign (Goldkuhl, 2009).

Service is defined differently in different areas. Sometimes the term is used to indicate an action performed by someone, or a set of actions, described as Web services, or the result of some action, which is a change in an object or a person. It could be seen as a value of some action. In the public sector it sometimes denotes organisational actions (Ferrario & Guarino, 2008).

Longman lexicon of contemporary English (McArthur, 1981) gives the following explanation of the concept of service: “something done by one person for another; something provided regularly, the way in which one is served”. Hornby (Hornby, 1974) defines the concept of service as “act of serving ” where “to serve” means “to perform duties, to supply with something, be satisfactory for a need or purpose, act towards or deliver something”.

49
The organisation for the Advancement of Structured Information Standards OASIS defines the term ‘service’ as: “a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is consistent with constraints and policies as specified by the service description” (OASIS, 2006).

In economics, a service is often seen as a non-material equivalent of goods. It is described as a process that creates benefits by facilitating a change in customers, a change in their physical possessions, or a change in their intangible asserts. In comparison to goods (Vargo & Lusch, 2004, 2008) that are transactable and transferable, services are transactable but not transferable. According to Ferrari and Guarino (2008), services are not transferable because they are events not objects that can be transferable, and as events are not ‘ownable’ they cannot be transferable.

Very often the concept of service is attached to technology-oriented definitions, referring to Web services and Service-Oriented Architecture (SOA). In SOA, a service is: “an application function packaged as a reusable component for use in a business process” (Colan, 2004). It either provides information or facilitates a change in business data from one valid and consistent state to another. In the area of Web services, one of the definitions of ‘service’ is the following: “it is a location on the network that has a machine-readable description of the messages it receives and optionally returns” (Newcomer & Lomow, 2005). The definitions of Web services presented by the W3C glossaries provides different perspectives on the concept as “an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities” but also as “a software system designed to support interoperable machine-to-machine interaction over a network” (Ferrario & Guarino, 2008).

The concept of service in the area of IS development is mostly recognized for its important role in information technology projects (Johannesson et al., 2008) and is mostly bounded to the term of service-oriented architecture. Developing services and deploying them using service-oriented architecture (Erl, 2004, 2005) is the way to utilize technological solutions to meet business demands. According to Hagg and Cummings (2008), SOA is a software architectural perspective, where service is the same as component in component-based system development methodologies. It represents a set of guidelines and design principles such as loose coupling, minimization of dependencies, encapsulation, re-use and composability (Zimmermann et al., 2004) in which business processes can be effectively reorganized to support the business strategy (Papazoglou & van den Heuvel, 2006). It defines a software architecture that comprises distributed components which are cooperating through a communication channel, which enables the construction of composite applications (Glen & Andexen, 2007). From a business manager’s perspective, SOA can provide the possibility to reach business flexibility. It enables business processes to be analysed in terms of services (Erl, 2005).

Conflicting views on the concept of service is one of the obstacles to the attempts to develop a new science of services (Chesbrough & Spohrer, 2006) and new academic programs focusing on services (Alter, 2008). Service science is a new discipline that studies value creation through services from different perspectives such as technical, behavioural and social (Edvardsson et al., 2007; Nilsson, in press).
It is a discipline consisting of fundamental theories and methodologies dealing with service structure, service processes, service modelling and management (Song & Chen, 2009). This discipline takes a broader perspective of services as opposed to technical interface descriptions (Sheth et al., 2006). In the area of service science, the concept of service is defined as: “an application of competence (by service provider) for the benefit of another (service requester) and can be understood as an action, or a set of actions that are performed for some value” (Spohrer et al., 2007).

In the context of enterprise architecture and enterprise modelling, it is necessary to have a broader understanding and interpretation of the service concept as the definition of service goes well beyond services that are realized using software applications. The definition of a service provided by Sheth et al., (2006) emphasises a provider-client interaction that creates and captures value. It emphasizes an exchange between two or more parties and a transformation received by a customer (Chesbrough & Spohrer, 2006).

The convergence of information and communication technology (ICT) design creates new opportunities such as service-oriented thinking that could be used to effectively leverage the value of knowledge in service relationships that produce high business value (Demirkan et al., 2008). A service-oriented way of thinking should not be just technology-oriented as in SOA terms (Colan, 2004; Papazoglou & van den Heuvel, 2006). The main purpose of service orientation is to capture business-relevant functionality as a service and to provide detailed information that the customer can use (Weske, 2007). Taking into account the nature of the concept, which is based on interaction between different actors to create and capture value, a service-oriented way of thinking could be applied for a computation-neutral way of analysis and design of business process.

3.9. Service in Service-Oriented Analysis and Design

The proposed method in this thesis is based on the ontological perspective of the concept of service. The understanding of service is based on the analysis of the very nature of the concept. The ontological analysis and understanding of service also contribute to a better understanding of organisation or organisational semiotics (Gazendam, 2004; Gazendam & Liu, 2004; Liu, 2000; Stamper, 1973). Such analysis defines the main elements that compose the concept, and explain why and how these elements are related. The ontological foundation of service has no direct relation to technological solutions. The objective of service is to provide the description of what is happening, not being biased to implementation issues. It is not ‘the service’, which is delivered and has value to the service requester, but the value of the content (Ferrario & Guarino, 2008). This content cannot be achieved without the dynamic process, without an interaction between some actors, the result of which creates value to the actors (Gordijn et al., 2000; Gordijn et al., 2006).
From the service definitions mentioned above, it is possible to state that a service is, first of all, a dynamic act of doing something to somebody. It means that there are more elements necessary to construct a concept of service than just the process of ‘doing’. As there are always a few actors involved in the ‘doing’ process, it signifies that it is a communication act or an interaction between at least two actors, human, organisational or technical components: one actor who asks for a service and one who provides it. If one is doing something, it means some action takes place. The action is always purposeful or goal-driven and prescribes responsibilities for actors involved (Alter, 2008). The action, being goal-driven, should always result in some value to the actor; be it physical, informational, or emotional.

From this description, it is possible to conclude that service is a complex concept, having structure. The elements of this structure are acting together for the production of a desired result. To get the result, which provides some value on demand, four key elements are necessary: service requester, service request, service provider and service response. Without one of these elements, the concept of service loses its semantic meaning. From an IS analysis point of view service can be viewed as a function, which is defined by a number of flows into opposite directions between a service requester and service provider. Each service response (output) is a function of a service request (input). Service providers are actors that receive service requests and transform them into responses that are sent to the service requesters. It is an interaction loop between service requester and service provider. This idea is represented graphically in figure 4:

![Figure 4: Service as an interaction loop](image)

A service in service-oriented analysis and design could be defined as a collection of ordered and purposeful interactions between various enterprise actors. Service interactions can be conceptualized as various types of actions that take place between enterprise actors. Service architecture is a composition of various types of enterprise actors and a set of interaction loops between them. Actors that could be seen as external entities or active elements (Dietz, 2006a) are organisational and technical subsystems.

The major characteristics, which are important for the development of service-oriented modelling method, are as follows:

- enterprise is a system. Enterprise system is a composition of subsystems, which can be viewed as interacting components (Bunge, 1977). Every subsystem can play a role of service requester and service provider.
when subsystems interact, they cause changes in objects that are manifested by the properties (Evermann & Wand, 2009).

organisational and technical components are represented by loosely-coupled, subsystems, which are not overlapping in functionality. Subsystems can be decomposed and linked by service interactions. Loose coupling is one of the main principles of service orientation (Erl, 2005).

Any interaction loop between actors must be motivated by the resulting value flow. In this way, an enterprise system can be defined by using service architecture, which is represented as a set of loosely-connected interacting subsystems. Subsystems can be viewed as technical or organisational components. Organisational components can be individuals, companies, divisions or roles, which denote groups of people. Technical components are subsystems that can be represented by software and hardware.

3.10. Characteristic Features of Service-Oriented Analysis and Design

Service-oriented business engineering is a new emerging approach that has evolved from the object-oriented (Blaha & Rumbaugh, 2005) and component-based software engineering (Szyperski, 1998). Experience from service-oriented architecture implementation projects (Zimmermann et al., 2004) states that traditional information system modelling methods cover just a part of the required modelling notations that are currently emerging under the Service-Oriented Analysis and Design (SOAD) approaches. The term of service-oriented analysis and design is often used in the context of SOA in connection with adding some innovations for service orchestration or service repositories. The idea of a structural composition of an enterprise system using the conceptualization of service-oriented architectures is not new. The problem is that traditional methods for information systems analysis and design methods do not provide much support for the engineering of service architectures, because it is not clear how the concept of service can be explicitly defined using semantic models.

In this thesis, service-oriented architecture is understood as architecture of business processes based on service orientation. Service orientation is an architectural style, an approach that structures business processes as compositions of services (Erl, 2004). Service-oriented business processes can consist of composed activities and each activity can be made up of any type of service. This approach should not be influenced by any technical solutions, as it is business, not platform that drives the functionality that the services enable. Models used to define different aspects and perspectives of service architectures should be computation-neutral, because such conceptual representations are less complex and more comprehensible both for information system designers as well as non-technicians (strategy planners, owners,
users, system analysts), which play a key role in semantic information system integration and business system change management.

Service-Oriented Analysis and Design (SOAD) (Zimmermann et al., 2004) in this thesis, is based on the assumption that business process models are composed of loosely-coupled components, which are viewed as service requestors and service providers. They can be humans, organisations as well as technical components (Gustas & Gustiené, 2009b), that interact with each other to produce some value. Business processes of organisations can be analysed as services or a composition of services, and changes can be made by replacing or recomposing existing services. To achieve ultimate flexibility, this idea could be used on all levels of enterprise architecture. Characteristic features of SOAD that are of great interest in this thesis are presented in table 4.

Table 4: Characteristic features of SOAD

<table>
<thead>
<tr>
<th>Model Types</th>
<th>Modelling Foundation</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation Independent Model</td>
<td>Business-Oriented Analysis</td>
<td>Pragmatic level</td>
</tr>
<tr>
<td>(CIM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service-Oriented Analysis</td>
<td></td>
<td>Semantic level</td>
</tr>
<tr>
<td>Platform Independent Model</td>
<td>Component-Oriented Design</td>
<td>Syntactic level</td>
</tr>
<tr>
<td>(PIM)</td>
<td>Object-Oriented Design</td>
<td></td>
</tr>
</tbody>
</table>

Service-oriented enterprise architectures allow for defining business processes in terms of more specific services. At the bottom level of abstraction, a service is an implemented business process function that is packaged as a reusable software component. Models of the syntactic level should define implementation-oriented details, which explain the data processing needs of a specific application or software component (Davis & Olson, 1985). All three SOAD levels are interrelated. The pragmatic level prescribes and motivates the static and dynamic architecture of services at the semantic level. It provides motivation behind interaction, structural and behavioural aspects of services. Semantic models of enterprise can be analysed in the context of pragmatic patterns (de Moor, 2005), which are refined in terms of communication actions (Dietz, 2001). Implementation related details of software components, which can be defined at the syntactic level by using object-oriented principles, are constrained by enterprise models at the semantic level.

Service orientation in business design should guide engineering of supporting IT, manage commitments for achievement of customer satisfaction (Denning & Medina-Mora, 1995). Service orientation, which is characterized by computation-independent models and design principles, enables enterprise systems to be analysed as service compositions. Service-oriented analysis could be used for motivation of technical components used and to get a holistic understanding of how the overall business process is composed. The principles of service-orientation can be successfully used for the separation of concerns by breaking down the enterprise
system into coherent subsystems, which are represented as a set of service requesters and service providers. The ability to conceptualize emerging configurations of service architecture provides significant competitive advantage; as such analysis and representations are more understandable for business managers that play an important role in overall system specification integration.

3.11. Desirable Characteristics for Service-Oriented Representations

Service-oriented diagrams that are used to represent IS solutions on different levels of abstraction should be evaluated by certain criteria (Gustas & Gustiené, 2004; Gustiené, 2003b). These criteria are important for identifying undesirable characteristics such as inconsistency, incompleteness, incoherence, redundancy and ambiguity, which lower the semantic quality of IS specifications. Semantic quality is essential when the modelling product is intended to be used as a means of communication of various architectural solutions among different stakeholders.

Identification of undesirable characteristics in diagrams is critical at the semantic level. Semantic representations are describing and constraining implementation-oriented diagrams at the syntactic level. They also refine diagrams that are defined on the pragmatic level. On the other hand, pragmatic representations should justify enterprise models at the semantic and syntactic level. An integrated and systematic way of dealing with graphical dependencies at various levels of abstraction might help to use enterprise models to identify such undesirable characteristics of IS specifications, which can then facilitate a better understanding of quality problems during the enterprise modelling process.

The enterprise models can be used as a uniform basis of reasoning about inconsistency of business process diagrams that are defined on the various levels of abstraction. It is not easy to attain semantic consistency due to a natural variation between views of the same business activity. Consistency between a more specific and a more general business process diagram can be controlled on the basis of a special set of inference rules. Inference rules may help to discover inconsistencies between dependencies on different levels of abstraction (Gustas, 2005, 2010). According to the service-oriented modelling method, a more abstract level is constraining a more specific one. This would imply that a more abstract diagram is a composition and generalisation of a more specific diagram.

Semantic differences of conceptual views are typically introduced by the different visions of stakeholders. A key problem is to determine the true needs and how these needs can be integrated into unambiguous, complete and consistent graphical descriptions of a system. Inconsistencies between constructs of diagrams that are represented on two neighbouring levels of abstraction indicate that the two diagrams are incompatible. Inconsistency demonstrates that the semantic descriptions of enterprise components are contradictory. Consistency issues have to be resolved before the engineering of any software component can take place. Changes in
organisational or technical components might be a reason why new consistency problems need careful analysis. Technical and organisational component fitness is one of the main issues that can be studied by using service-oriented modelling method.


together analysis means questioning the integrity between the computerized information system state change and communication flow’s internal structure. Any parameter of a communication flow is supposed to be used for a certain purpose. It might be a creation or termination of an object in a certain state. Termination action in a specific state must destroy all association that is relevant for this state. A creation action must establish all relevant associations that are specified in the diagram, otherwise the action is rejected. Communication flow parameters can be consumed or emitted by a predefined action. A consumption action is supposed to store parameter values into the post-condition class attributes. State is defined as a collection of attribute values. Information flow parameters have to be defined in a post-condition class. If a flow parameter matches different name of the attribute, then an explicit computation rule for this attribute must be defined. Usage of a pre-condition and post-condition class attribute should be questionable with respect to an overall structure of incoming flow. If an attribute is irrelevant for the action, then it should be removed. Any object can be terminated by a removal action, in which case no post-condition class attributes should be specified. The termination process can be represented by an action without a resulting class. In this case, no commitments of actors should be pending in connection to this state. A properly designed termination action should always emit an information flow that represents a collection of the deleted attribute values. Such a flow can be viewed as a confirmation message to the initiator of action.

Completeness, in general, refers to whether all the elements necessary for the system to be modelled are included. The semantic completeness of actions in all resulting branches is always relative. However, every service interaction loop must be consistent with respect to some design goal. Goals define the system boundary and therefore may help designers to control a relative pragmatic completeness of a specific business process. A complete business process should consist of coherent actions and all service interactions. Service interaction loops cannot be broken, as it results in discontinuity problems.

Semantically incomplete representations of business processes can be characterised by the presence of semantic holes. The semantic holes typically appear when inapplicable dependency is used. That is why the service-oriented modelling method proscribes using optional dependencies at very late analysis and design modelling phase. A completely applicable set of dependencies cannot guarantee a completeness of representation. Nevertheless, a semantically complete specification must always be hole-less. A totality of the static dependency would indicate that for any instance of one concept there is always an association to an instance of the dependent concept. The totality of the state transition dependency means that any object, which belongs to a pre-condition class, is applicable for the specified transition link. The totality of the actor dependency would mean that any instance, which is classified as a service action.
requester, is privileged to send a request to a service provider by initiating an action, if certain pre-conditions and post-conditions hold.

Semantic redundancy in general terms would mean that two different sets of semantic dependencies are used at the same time to express the same meaning. A special set of inference rules can be used to discover redundancy (Gustiené, 2003b). If the derived dependencies were represented in a same diagram, then they would indicate a situation of semantic redundancy. Such an undesirable characteristic hinders the effective communication among the members of an enterprise engineering team.

Ambiguity of concepts in the modelling process indicates a low semantic quality of IS specification. Very often it is due to the incompleteness and misunderstanding of business processes. Ambiguity can arise because systems are changing over time and spanning across organisational and technical system boundaries. Ambiguity of the meaning can occur because the message is vague and inconsistent in the context the message belongs to. Synonyms and homonyms of the concepts also can cause ambiguity. The service-oriented way of modelling is useful for identification of undesirable system characteristics and enables improving the quality of system specification. Enterprise models and semantic dependencies used are intended to support reasoning and validation of IS specifications before they are implemented.
Chapter 4
Service-Oriented Modelling Method: Modelling Process and Language

4.1. Modelling Process

Modelling process is one of the corner-stones of the presented service-oriented modelling method. It is a designed and determined sequence of activities and organisational procedures that will enhance collaboration in the development team and contribute to the delivery of a quality product to the customer. It includes modelling activities and techniques through three levels of abstraction and defines the integrated order of behaviour during the entire modelling process. The modelling process facilitates a coherent transformation from one level to another. Every process is goal-driven and is intended to have an initial and final stage.

System analysts in the area of enterprise engineering first define a very general and ambiguous business process at the pragmatic level. This scenario is similar to the process of system conception (Blaha & Rumbaugh, 2005), the purpose of which is to defer details and understand the big picture of what the future system is. Later, system designers gradually extend analysis representations by placing numerous assumptions into well-defined information system representations on the semantic level. To reach a consensus, the analysts integrate various parts of IS specifications and play scenarios with objects of various types, in order to validate and justify technical system components at the syntactic level. Top-down analysis facilitates a better semantic quality of enterprise engineering (Gustas & Gustienė, 2004). The modelling process includes a set of rules, which define how the process should generally be carried out. The design and architecture of the system embody the most important decisions about how the system is built and implemented.

The success of the modelling process depends on two things: on the modelling language used to form models and on the process of how these models are used. Modelling constructs should be communicated; therefore the modelling language should be expressive enough to convey the design. It should be unambiguous and easy to use so that it facilitates the identification of contradictions that cause communication problems. Usually, the graphical language used is diagram based, because natural language is too ambiguous and often causes misunderstanding. It is not enough to develop a modelling language for graphical representations. The entire modelling process should be provided by an appropriate modelling way that guides the modelling process in a systematic manner, which contributes to a better quality of communication among stakeholders.

The service-oriented modelling process proposed in the thesis has the following
advantages:

- it guides the sequence of activities through three levels of abstraction and provides a way of bridging from pragmatic to semantic system representations, as well as mapping from computation-neutral modelling to implementation-specific design,
- service-oriented analysis helps to graphically represent the interplay among interactive, behavioural and structural aspects of IS specifications,
- the semantic dependencies of service-oriented diagrams help to detect undesirable characteristics such as inconsistency, incompleteness, redundancy and ambiguity of implementation-specific modelling.

The success of service-oriented analysis and design depends greatly on finding the appropriate fitness between the three levels of the enterprise framework, which is represented in figure 5.

![Figure 5: Architectural framework for service-oriented modelling](image)

The fitness of system specifications between these levels is critical for the success of the final result. The fitness and consistency between levels depends much on having techniques for the refinement of pragmatic entities (goals, problems and opportunities) that represent their structural and dynamic aspects at the semantic level. The methods should provide a guide to support the development and transition from the early requirements to the design of the system. Conceptual models require a considerable effort to be mapped to software artefacts (Perrone et al., 2005a). An integrated modelling way provides the guidelines for the transition between levels. It also provides a way to map from computation-neutral to implementation-specific modelling. The transition possibility enables a check consistency between levels. It also underlines the possibilities for requirements
traceability, which is a critical part for change management (Maciaszek, 2005), verification and validation. Verification refers to a process of ensuring that the design or implementation conforms to the requirements (Leithridge & Laganière, 2005). Validation refers to an activity meeting the requirements agreed upon at the analysis phase and ensure its quality (Kamal, 2008). It refers to the activity that enables system designers to check if analysis and design really satisfies the criterion stated by the external user of the service. Semantic dependencies used during the modelling process provide the possibility to control the semantic modelling quality. It provides the new ways of semantic incompleteness control (Gustas & Gustiené, 2004; Gustienė, 2003a).

The architectural framework of the service-oriented modelling method is similar to the MDA framework, as it also provides computation-neutral and implementation-specific modelling that corresponds to MDA computation independent model and platform independent model. The difference is in the purpose of the framework built in modelling language and techniques. The purpose of the MDA framework is to support a model-driven engineering of software systems. This framework uses UML and other OMG modelling standards. The service-oriented modelling method is developed for information systems analysis and design, with a focus on the conceptual modelling of IS specifications. The service-oriented modelling method is based on a modelling language and a set of modelling techniques for specifications of early requirements, where pragmatic and semantic aspects of service architectures can be analysed together. It uses language that is based on interaction principles and that uses a single type of diagram to present structural and behavioural aspects. Additionally, the method provides a modelling process that supports a traceable way of refining pragmatic specifications into conceptual representations of IS. It also provides transition principles from computation-neutral to implementation-specific modelling.

Computation-neutral representations of enterprise architectures (Finkelstein, 2004) that can be used by non-technicians are a major focus in this study. Models at the computation-neutral modelling level play a key role for semantic system integration in a business-oriented (supports a planner view) and service-oriented analysis (supports an owner view) phase. Computation-neutral models define the system from the computation-independent viewpoint, which is necessary for bridging the gap between business analysts and system design experts. They should be established before any implementation-specific decisions are taken. It is recognised that UML support for such a task is quite vague (Perrone et al., 2005b), because integration semantic principles of different diagram types are still lacking (Harel & Rumpe, 2004).

In the service-oriented modelling method, modelling done at the pragmatic and semantic levels is computation-neutral. Semantic and pragmatic models define only essential aspects of service architecture and are built on an integrated set of semantic dependencies. The semantic level is guided by the pragmatic principles that are placed on a higher level of abstraction. It defines constraints for the syntactic level as well as it provides the solutions for the pragmatic level. Coherence between the
semantic and syntactic levels is crucial for the success of the final implemented product. Models created at every level are necessary for the identification of inconsistencies of the analysis process. They provide the identification of inconsistencies at an early analysis phase, which contributes to the possibility to react in a timely manner. Coherence of the models used on different levels is necessary to make informed business decisions and create value for the enterprise (O’Rourke et al., 2003). The implemented product should be aligned with the goals stated by business analysts and this depends on reaching consistency between computation-neutral and implementation-specific modelling.

The modelling process is pragmatic-driven. A top-down analysis begins at the pragmatic level, where business goals and objectives are discussed. This level is important, because here, the pragmatics of services in terms of goals is discussed. Understanding the goals is essential to guide the design of the business processes and to evaluate the operational quality of the solutions taken by business experts (Gao & Krogstie, 2009). The analysis of goals is important for semantic completeness control as well as for determining the scope of the business process boundaries. By focusing on service architecture, in terms of how business experts understand it and not concentrating on technology thinking, make service-oriented analysis and design more profitable.

Pragmatic specifications done at this level are strategy-oriented descriptions, which drive the semantic analysis further. They provide the motivation for organisational and technical system components that can be viewed as interacting actors of an enterprise. The advantage of a pragmatic-driven approach is that it provides way of systematic transition from the pragmatic level, where goals are discussed and pragmatic specifications produced, to the semantic level, where structural and dynamic aspects of service will be analysed. It provides a way to show how business processes as services, discussed at business-oriented level, will be refined in terms of structural and behavioural aspects.

At the semantic level, the integration of static and dynamic aspects is performed, which is possible using service-oriented constructs. Conceptual modelling done at this level provides an overview of how various activities of the organisational system, in terms of services, are organised and how they are related. They should help business managers to understand the proposed solutions as well as to communicate possible ways for how the IS can be built.

Service-oriented diagrams provide a foundation for logical design at a syntactic level, the level where implementation-specific details are taken into account. The alignment of conceptual representations with technology-oriented design is necessary, as every goal should be consistent with the final implemented product. Implementation-specific modelling can be demonstrated by using UML. Motivation to it is my experience in teaching object-oriented analysis and design courses. Working with students that use object-oriented approach and UML as modelling language in their practical work, and seeing some difficulties, gave me confidence in my research. Using different types of diagrams to represent different aspects of IS creates difficulties in reaching consistency and traceability between the diagrams.
Service-oriented modelling considers a twofold nature of IS analysis and design process: among different aspects of EA and different views of stakeholders. Such a modelling process is particularly flexible for introducing evolutionary changes of IS specifications. This integrated and systematic way of service-oriented analysis contributes and provides the guiding principles from goal modelling to service-oriented specifications and to implementation-specific design. The advantage of such a process is that it can help the IS designers to focus on computation-neutral aspects of IS system rather than just on technology-related solutions. This method is built on a service-oriented modelling process, which is pragmatic-driven and computation-neutral. The purpose of such a modelling method is to provide a clear system of decomposition principles as well as principles for the separation of crosscutting concerns. These principles are critically important for management of IS complexity and integration issues as well as for minimizing inconsistencies and discontinuities of business processes that may cross boundaries of enterprise, when some changes take place (Papazoglou, 2008).

4.1.1. Pragmatic Dependencies

A business-oriented analysis done at the pragmatic level provides the motivation behind new business solutions. The business strategy of an organisation defines the direction and scope of possible changes in the organisation and defines the organisation’s position, which is important for a competitive advantage (Foolthuis et al., 2008). This is the first and the most abstract step, where goals are stated and discussed. Strategic and operational choices taken at this level are key in determining how the work process change will be approached (Rouse & Baba, 2006). This level concentrates on a strategic description, that is supposed to give a definition of the ‘why’ - a long term intention or a vision of the enterprise under development.

Any business process can be viewed as a collection of services. From the pragmatic point of view, service functionality can be regarded as a problem, opportunity or a goal. Service functionality can be desirable or not desirable. Pragmatic aspects are driven by the goals, problems and opportunities of information systems designers. An analysis of services can help business experts to analyse possible changes for creation of additional business value. At this level, a service is characterised in a prescriptive way (Ferrario & Guarino, 2008), where the ‘rules of the game’ are established. Here, the types of actions composing the service, the actors and their responsibilities, are analysed. The responsibilities are usually assigned according to specific structural and behavioural constraints in the organisation.

Pragmatic dependencies are used to define the intentions of the actors involved in the business processes (Gustas, 1998, 2000; Gustas & Gustiènè, 2002, 2004). They can also be viewed as a modelling basis to reason about the intentions of designers on new system architecture. Graphical notation of the pragmatic dependencies is presented in figure 6.
### Pragmatic Dependencies:

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal Dependency</td>
<td>$g$</td>
</tr>
<tr>
<td>Problem Dependency</td>
<td>$p$</td>
</tr>
<tr>
<td>Opportunity Dependency</td>
<td>$o$</td>
</tr>
<tr>
<td>Negative Influence</td>
<td>$(\cdot)$</td>
</tr>
<tr>
<td>Positive Influence</td>
<td>$(\cdot)$</td>
</tr>
<tr>
<td>Refinement</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6:** Legend of the pragmatic dependencies

- **Goal dependency** is used to specify either the desirable situations or desirable structural properties in a system. It links enterprise actors to the intentional notions such as goals or objectives.

- **Problem dependency** is used to specify undesirable situations. Restrictive notions such as problems or causes of problems express either not desirable properties of a system or undesirable situations that enterprise actors are intended to avoid. Problem typically hinders the achievement of a goal. Opportunity dependency can be used to refer to future situations that can be taken advantage of to improve a current situation. Opportunities can also be viewed as strength in the existing system or a business process. An opportunity is a desirable fragment of an existing semantic specification that is intended to be maintained in a new system. The desirable features of a problematic situation can be indicated as opportunities by using refinement links.

- **Negative influence dependency** ($\cdot$) and **positive influence dependency** ($\cdot$) are used to denote influences among goals, problems and opportunities. Negative influence dependency from A to B indicates that A can be regarded as a problem, because it hinders the achievement of goal B. The positive influence dependency from A to B would mean that A can be viewed as an opportunity for the achievement of goal B (Gustas et al., 1996).

- **Refinement dependency** is used as a means of pragmatic decomposition of goals, problems and opportunities. Pragmatic entities are typically refined by referring to other more specific situations.

Goals of various organisational components stimulate interaction among actors, leading to further interactions (Warboys et al, 1999). Goal analysis has substantial promise in aiding the elicitation and elaboration of requirements (Mylopoulos & Yu, 1998). At the same time, goal analysis provides the control of commitments that different actors are suppose to fulfil (Shahzad & Zdravkovic, 2009). Any actor can achieve a goal by avoiding a problem. Goal analysis is a useful tool to deal with various kinds of conflicts between pragmatic entities (Gustas & Gustiené, 2008), as
the meeting of one goal may interfere with others goals (Mylopoulos & Yu, 1998). It should be noted that the interpretation of requirement as a problem, opportunity or goal is relative. The achievement of a goal by one actor can be regarded as a problem for another actor. The goal, opportunity and problem dependencies can be used to refer to desirable or not desirable situations. Goals or problems can be decomposed by using a refinement link.

An important part of the business process analysis is the way in which different goals, opportunities and problems are related. Pragmatic decomposition helps business designers to see the deeper structure of the pragmatic entity. It can also be used as a means of eliminating problematic or conflicting fragments of specification at the semantic level and complimenting them with desirable fragments. The hierarchical structure of pragmatic entities determines the strategies and the ways in which the high-level objectives are achieved. Pragmatic entities can be decomposed using more specific elements. Goals can be refined into sub-goals and problems can be refined into sub-problems. Refinement dependency is similar to decomposition dependency, but the difference is that in the case of decomposition, the ‘whole’ is decomposed into all possible sub-parts; it can be called complete decomposition. In case of refinement dependency, the ‘whole’ is refined just into those sub-parts that are of great interest or importance in that specific situation.

Refinement of A into B is denoted be B○ A. It states that B satisfies A, where B can be partially used for the achievement of A. The refinement link is characterized by the following condition (Gustas & Gustiené, 2008): B ○ A if and only if S(B) ⊆ S(A) and P(A) ⊆ P(B), Here: P(A) and P(B) respectively denote sets of pragmatic dependencies of concept A and concept B, S(A), S(B) respectively denote sets of the semantic dependencies that define pragmatic entity A and B. The presented formal condition (Gustas et al., 1996) consists of two parts:

1. A set of the semantic dependencies that define pragmatic entity B is a subset of a set of the semantic dependencies of A, and
2. A set of the pragmatic influence dependencies of entity A is propagated to B.

The second part of the condition is defined as follows:

if B ○ A and A → C then B → C,
if B ○ A and A ← C then B ← C.

Figure 7 presents the example of goal decomposition.
In the example above, the goal of Travel Agency, ‘To Propose Travel Services’ is refined into two sub-goals ‘Selling Trips’ and ‘Limousine/Car Services’ (incomplete decomposition). Sub-goal ‘Selling Trips’ is decomposed into two sub-goals ‘Processing Trip Reservation’ and ‘Make Payment’ (complete decomposition). Sub-goal ‘Processing Trip Reservation’ is further decomposed into four sub-goals (complete decomposition) necessary to achieve the sub-goal ‘Processing Trip Reservation’.

Influences between pragmatic entities can be analysed using negative or positive influence dependencies. Problems usually negatively affect reaching goals. One way to eliminate this problematic situation is to weaken a problem by introducing a new opportunity, which negatively influences a problem and positively influences a goal. This situation is represented in figure 8.
Problem ‘Sharp Competition among Travel Agencies’ negatively affects the goal ‘To Propose Travel Services’. A new introduced opportunity ‘Discount for Returning Customers’ is supposed to negatively affect the existing problem and positively affect the goal ‘To Propose Travel Services’.

Pragmatic specifications aim to provide the motivation for conceptual representations of enterprise components, which are defined in terms of interaction loops between service requesters and service providers at the semantic level. An analysis of goals is crucial for the final result, because the underlying assumption is that system services are worthwhile if they meet the goals of the organisation. Pragmatic specifications can be used as a driving force for the analysis of the semantic structure of service representations.

4.1.2. From Pragmatic Specifications to Semantic Structure of Service

The semantic (conceptual) level has to have the capacity to describe the static and dynamic aspects of business processes across the organisational and technical system boundaries (Gustas & Gustienė, 2004). Models defined at this level show the activities performed by various actors. Their activities are usually prescribed by goals. The semantic level may provide the description of service interactions with pre-condition and post-condition classes. It defines the semantics of the concepts and relationships used, which define the basic content of a business process (service or composition of service). Modelling at this level should also have the capacity to show the decision making process, resulting in the selection of a course of actions among several alternatives. If the overall system behaviour is specified, it is also possible to express alternatives, sequences and synchronisation actions.

The semantic specifications can not be analysed in isolation from the pragmatic specifications. This way of modelling provides a foundation for checking coherence between levels. At the lower level of abstraction, it is necessary to refine the pragmatic entities into services, which can be viewed as fundamentals of service-
oriented way of modelling. After stating goals and analyzing the possibilities and solutions at the pragmatic level, it is important to show how these goals are refined further at the semantic level. Refinement of pragmatic entity (sub-goal) into interaction loops is represented in figure 9.

![Figure 9: Refinement of pragmatic entity](image)

This figure presents just one refinement of a sub-goal and just one interaction loop. It does not show how this sub-goal is integrated in the whole description of the service ‘Search Trips’. All four sub-goals presented in the example are sub-goals of the goal ‘Selling Trips’ and they all should be refined into services and integrated further in a single diagram, representing the whole business process of the service ‘Selling Trips’, including the static and dynamic aspects. This will be explained further in the thesis.

There are two major classes of semantic dependencies used at the semantic level: static and dynamic. In enterprise modelling, all kinds of conventional static relations can be defined between concepts such as classes, actors, states and flows (Gustas & Gustiené, 2002). The semantics of static dependencies can be defined as cardinality constraints (Blaha & Rumbaugh, 2005; Maciaszek, 2001; Martin & Odell, 1998).
Dynamic semantic dependencies are used to define relations between different actors, their actions and communication flows (Gustas & Gustiené, 2002). The set of static and dynamic semantic dependencies plays an important role in defining interactions between service requesters and service providers. Understanding interactions is critical in various business modelling approaches, because they serve as a basis to analyse obligations and authorisations among actors involved. A typical action workflow loop can be defined in terms of two or even three interaction dependencies between two actors. Interaction dependencies can be used to explore the opportunities that are available to these actors. The complete explanation of semantic dependencies will be presented further in the thesis.

4.1.3. From Actors to Syntactic Elements

From a software designer’s perspective, it is not enough to represent the system architecture from a pragmatic and semantic perspective. System designers need to understand a technical system architecture, where the application is going to be installed. The modelling language should be able to graphically represent the alignment between organisational and technical system boundaries. The syntactic level should define the implementation-specific details, which explain the data processing needs of a specific application or software component (Davis & Olson, 1985). Implementation is understood to be assignment of technological means to the elements in the implementation model, so that the system can operate (Dietz, 2006b). Enterprise actors, which at a semantic level are represented by square boxes at the syntactic level, can be thought of as humans or technical components. Enterprise system architecture can be designed as a decomposition of loosely-coupled files, software or hardware components. Figure 10 represents the notation of components that can replace the actors, used in various projects.

Figure 10: Notation of components
Actors can be decomposed into organisational and technical components. Organisational subsystems can be represented as humans or business components. Classes and data files may represent databases, data warehouses or other types of data storage. Typically, a coherent set of interactions are delegated to one independent component. Service interactions can be defined in terms of components and their interfaces with the structural definition of the printout, message, screen or file layouts. They are supposed to define existing or expected interaction flows to support the intentions of various actors (Gustas & Gustiené, 2004). Screen and printout forms should be customised for each interaction flow between organisational and software components. Message forms are designed for the implementation of interaction flows between software or hardware components. The notation for interfaces between technical and organisational components is represented in figure 11.

![Figure 11: Notation of interfaces](image)

A service loop can be defined in terms of technical components and organisational components and interfaces between them. A short illustration is presented in Figure 12.

![Figure 12: Interfaces between technical and organisational components](image)

SF1 is the screen form of Flight Requirements and SF2 is the screen form of Flight List. The graph defines explicitly the infrastructure of the Internet Agent (IA) component. It shows user interface with a software component. When the User
begins the process, the IA software component invokes the action *Show Flight Requirements Form* and SF1 screen form is displayed to the User. When the User fills in *Flight Requirements* and clicks the button *Search Flight*, the operation *Search Flight* is invoked and the IA component is searching for the flights. If the matching flights are found, then the IA invokes operation *Show Flights* and the SF2 screen form *Flight List* is displayed to User.

All coherent interactions that fit together for the achievement of service requesters' goal can be implemented as autonomous software/hardware component. Therefore, the identification of interaction flows is crucial for breaking down the enterprise system into coherent non-overlapping services. Every enterprise may define its own set of symbols used for the representation of various component types. Syntactic elements are considered as the implementation perspective or CASE tool dependent. For example, database relations are typical representatives of the syntactic level. File or database layouts can be defined by using the conventional programming languages or traditional database definition languages. Almost the same set of syntactic elements could be used for the object-oriented approach. In this case, the notion of a file layout should be replaced by the notion of class layout, which would contain the operations as well. The generalization and aggregation hierarchies of actors, in terms of the technical components that are used in different communication loops, would define a relevant hardware or software system architecture. Other types of icons, such as fax machines, phones, and computer networks can be introduced on demand. Syntactic elements could be seen as building blocks for the representation of technical deployment architecture, software component infrastructure or organisational dependency structure (Gustas & Gustiené, 2004). An implementation-dependent file, message, printout or screen layout is supposed to define the technical architecture. All three levels of the framework are interrelated as they define the same artefact. Diagrams at the syntactic level are constrained by the dependencies at the semantic and pragmatic levels.

Modelling and design process through three architectural levels provides the techniques and guidelines for the transition from one level to another. This enables possibilities for the validation of goals stated at the pragmatic level and the verification of logical design made at the syntactic level to conceptual modelling done at semantic level. It provides a way for semantic traceability via all three levels and enables interplay between business requirements and technical solutions, which provides the natural view to understand the modelling artefact as a whole.

### 4.2. Service-Oriented Modelling Language

This section presents the second corner-stone for the service-oriented modelling method. It defines the basis of the modelling language used for the service-oriented modelling method and explains the idea of flexible interpretation of concepts and
presents the notation used for service-oriented analysis and design. It presents a meta-model that defines the elements of modelling language. The elements are actors, concepts, actions and dependencies that are both static and dynamic.

Modelling method and modelling language are two important contributions of this work. A method is an explicit way of modelling, consisting of language notation, modelling techniques and guidelines. It is important because it predetermines a set of modelling activities, telling what to do and why. A modelling language is necessary for building service-oriented diagrams. Modelling elements and notations are important working techniques for the stakeholders involved in the information system development process. They serve as the foundation for understanding and managing business processes and the aligning business with information technology architectures. Modelling notation provides the foundation for specifying the internal and external service behaviour in a graphical way.

4.2.1. Interpretation of Modelling Concepts

The interpretation of the modelling concepts for the representation of a problem domain is very important (Boman et al., 1997) as the concepts determine the semantic correctness of requirement specification. Requirement specification process concerns the modelling of stakeholders’ requirements defined during the requirements determination phase (Maciaszek, 2005). The content of the requirement statements plays an important role for expressing static and dynamic relations among concepts in a problem domain. The analysis of concepts is important as they are the objects and information holders of a problem domain, which are of the main interest (Gustafsson & Höglund, 2009). The interpretation of the concepts used in service-oriented modelling is relative. The motivation for the importance of concept relativity is to avoid structurally different but semantically equivalent representations (Gustas & Gustiené, 2002). Strict interpretation of roles is one of the main sources of difficulties in the area of view integration (Batini et al., 1986). Semantically equivalent, but structurally different, representations violate the independence principle - a main principle of conceptualization (van Griethuysen, 1982). For example: ‘to order’ and ‘to search’ are verbs, and ‘order’ and ‘search’ are nouns. Figure 13 demonstrates that the same concept can be interpreted in many ways.
Every noun in the model is captured by a box and a verb is captured by an ellipse. Action names are verbs, which bind together actors (service requesters and service providers). Any concept can be viewed as an actor, an attribute, or a class, depending on the combination of semantic relations that connect it to other concepts. Whether a concept is regarded as an instance, class, attribute, relationship, flow or an actor depends upon types of the static and dynamic dependencies through which they are related to other concepts. Possible interpretations of concepts are represented in figure 14. Semantic dependencies define how a concept can be classified.

A meta-model is a model of models. According to the FRISCO Report (Falkenberg et al., 1996), a meta-model is a model of the conceptual foundation of a language, consisting of a set of basic concepts and rules that determine the models in that language. It refers to domain-independent abstractions and demonstrates that any concept can be interpreted in different ways. It contains types whose instances are also types (Martin & Odell, 1992). This diagram should be analysed together with the meta-model of relationships. A meta-model of various types of relationships is presented in figure 15.
Figure 15: Meta-model of relationship

Relationship is one of the key constructs used in conceptual modelling. It is the most important element in most models used to capture how concepts are related to each other (Chen, 1976). Relationships between concepts are of two types: static and dynamic. ‘Action’ and ‘Rule’ represent two subsets of the ‘Dynamic Relationships’. Actions are defined in terms of interaction dependencies between different types of actors. Depending on the nature of a flow element in action, the communication flows can be information, decision or a physical flow. Flows represent messages or physical objects to be transferred from agents to recipients.

It should be noted that in the presented service-oriented modelling method, such concepts as relationships or association ends are avoided, because modelling in such a way restricts the specification evolution perspective. It means that during the modelling process, it may be necessary to capture and specify the properties that are represented by these association roles, which are unstable. The phenomena of semantic role relativity (Gustiené & Gustas, 2002) asserts that it is not possible to put a strict classification on the semantic roles the concept plays. The interpretation of concepts much depends on a modelling situation. The traditional modelling approaches do not support the semantic role relativity and it is a cause of various view integration problems.
4.2.2. Notation of Static Dependencies

Semantic dependencies used at the semantic level can be of two kinds; static dependencies and dynamic dependencies. Static dependencies constitute the basis for understanding the behaviour of information system objects. They are traditionally used to present only the structural aspects of concepts. The entire dependency notation used in service-oriented modelling is defined in the following section.

*Classification* is a static dependency. It can be used to define objects, which are instances of concepts. An instance can be viewed as an element of a set that is defined by a concept it belongs to. Sometimes classification dependency is referred to as instantiation, which is a reverse of classification. In object-oriented approaches, it represents a semantic relation between an object and a class. Instances of concepts propagate to the more generic concepts in the inheritance hierarchy. A graphical notation of the classification dependency is presented in figure 16.

![Classification dependency](image)

**Figure 16:** Classification dependency

An example of the classification dependency could be the following:

Honda CRX ← Product

*Inheritance* is a static dependency which is used for sharing similarities from more general concepts. It predefines the inheritance of static as well as dynamic dependency links for more specific concepts. The inheritance dependency is a very useful relation for reasoning about the semantic integrity of conceptualizations. It should be noted that in an object-oriented approach (Booch et al., 1999) inheritance is applied only for attributes and operations. In the presented service-oriented modelling method, the attribute, composition, interaction and transition dependencies are also inherited by the more specific concepts from more general ones. Inheritance is used for the specification of sub-concepts or super-concepts. More specific classes inherit the mandatory attributes and interactions of more specific classes. A graphical notation of the inheritance dependency is presented in figure 17.
An example of the inheritance dependency could be the following:

Student → Person

Inheritance is a relationship as a whole from other concepts that might be viewed as parts. In the presented service-oriented way of modelling, a stronger form of aggregation is used, namely composition dependency, which differs from object-oriented composition (Maciaszek, 2001). It allows just 1:1 or 1:M cardinality between a part and a whole. M:N cardinality is not allowed. This is characterised by the following properties: a part cannot simultaneously belong to more than one whole of the same type. If it does belong, then it must be the same whole. A part and a whole are created at the same time, once a part is created, it can be terminated at the same time the whole is terminated. It means that the life span of instances for compositionally dependent classes must be identical. Composition dependency can be used for reasoning about system components. An important characteristic of composition dependency is the possibility to propagate interactions from parts to compositional wholes. The interaction dependencies are propagated upwards in the decomposition hierarchy. It should be noted that derived interaction dependencies are redundant and they should not be represented together with basic links. Composition is both an asymmetric semantic relation inheritable by more specific concepts, and a transitive semantic relation. The importance of composition dependency is that it represents the related classes of objects, which must be synchronically removed or created when an action takes place. A graphical notation of the composition dependency is presented in figure 18.

An example of the composition dependency could be the following:

Order Product → Order
**Attribute Dependency.** Static differences in various classes are represented in terms of dependent concepts that are connected by the attribute dependencies. The attribute dependencies stem from the traditional semantic models. Attribute dependencies are used to define an important semantic difference between concepts. Their semantics are defined by multiplicities, which represent a minimum and maximum number of objects in one class that can be associated to objects in another class. One significant difference between the service-oriented modelling notation and the traditional approaches is that the association ends are nameless. The dependencies are never used to denote the association mappings in two opposite directions.

In the service-oriented way of modelling, a set of all static association varieties is used to distinguish between attribute and non-attributes (Gustas & Gustiené, 2009b). Only five basic associations with mandatory constraints are recommended for the final service-oriented analysis and design phase, since associations with optional constraints give rise to semantic holes (Gustiené, 2003b), which cause incompleteness and ambiguity of conceptual models. According to the service-oriented modelling method, the concepts that are connected by optional associations have under-defined semantics. ‘Null values’ have no meaning (Wand et al., 1999). Since optional cardinality is used in object-oriented analysis and design, it is not forbidden to use in early stages of service-oriented analysis and design. A graphical notation of the attribute dependencies and their cardinalities is represented in figure 19.

![Diagram of Attribute Dependencies](image.png)

**Figure 19:** Graphical representation of attribute dependency

An example of the attribute dependency could be the following: Product→Title

The purpose of such a strict interpretation of class and attributes is to avoid undesirable problems of conceptual specifications. Optional properties should not be used if the diagrams are intended to unambiguously communicate the semantic details of IS specifications. A disadvantage of such a restriction is that the diagrams will increase in size, but the advantage is that the diagrams will be controlled and the semantics of the domain will be conveyed unambiguously. A set of totally applicable dependencies can be considered as a good foundation for unambiguous system analysis and design.
4.2.3. Notation of Dynamic Dependencies

Dynamic dependencies in the service-oriented modelling method are used to define relations between different actors, their actions and communication flows. Descriptions of organisational activities, as well as actors involved in these activities, are based on the dynamic dependencies. The dynamic part of the enterprise model can thus be represented by actions that use and produce various communication flows, and by actors that are responsible for the initiation of those actions. The dynamic relations are state dependencies and communication dependencies. The communication dependencies among enterprise actors are relevant for description of the ‘who’ perspective. If concept A is connected to B by a communication dependency, then A is an agent and B is a recipient. Depending on whether there are flow elements in the action or not, the communication flows can be information, material or decision.

Any flow dependency can be described in more detail by using a communicative action. The actor dependency is considered to be both an action and a communication flow (Goldkuhl, 1995). Cohesion of action and communication flow results in a more complex abstraction, which is entitled as a communicative action (Gustas, 2000; Gustas & Gustienė, 2009a; Gustienė & Gustas, 2008). In such a case, the flow dependency link between two actors specifies that the recipient depends on the agent not only by the specific flow, but by the action as well. Actions will be represented by an ellipse. A graphical notation of the dynamic constituents in a communication action is represented in figure 20.

![Graphical notation of dynamic dependencies](image)

**Figure 20:** Graphical notation of dynamic dependencies
Interaction is a dynamic relation (→). It is defined by a communication action between two active actors. It indicates that one actor depends on another actor. Interaction can also be considered as an action and a communication flow. At the same time, interaction is a strategic flow (Yu & Mylopoulos, 1994) dependency between agent and recipient. Actions can be triggered by organisational actors or technical components. Interacting components can be perceived as service requesters and service providers. Interaction dependency is important because it prescribes the responsibilities of actors involved in interactions. The initiation of actions depends on the actors’ goals. An intersubjective perspective, which is expressed by interaction dependencies between different enterprise components, such as actors, is a starting point in service-oriented modelling (see 5.2.5).

Transition (→) is a dynamic relation that, in service-oriented modelling, is used to represent the fundamental changes of objects that take place during creation, termination and reclassification events. In service-oriented modelling, it represents the objective perspective of service. They are normally considered as ‘how’ perspectives. Transition dependencies are used to represent internal changes of objects, when an action takes place. Actions indicate permissible ways for causing transitions by manipulating properties of objects in different classes. The changes can be traced by checking the changes in attributes that define the properties of the objects. These changes motivate the actions that take place during the interactions. If these are no changes in attribute values, then the usefulness of some specific actions should be questioned and checked. The pre-condition and post-condition class objects are characterized by attributes, which must be sufficient to understand the details of interaction effects. It is one of the aspects that differ in the service-oriented modelling approach. In object-oriented approaches, transitions are never associated with classes, only with states (Harel, 1987). The example from Employment Service in figure 21 demonstrates interaction and transition dependencies, and semantic differences that are characterized by attributes.

**Figure 21:** Interaction and transition dependencies
The main semantic difference between concept Applicant and Employee can be detected in attributes. When the Applicant is reclassified to Employee, two new attributes are added that are specific to Employee: Contract and Position.

Analysis experts should agree on the coherent set of service interactions, which are defined using static and dynamic dependencies. One of the goals of semantic dependencies used in conceptual modelling is to clarify the differences among concepts. The presented semantic dependencies can be used to specify the semantic details of process and data on various levels of abstraction.
Chapter 5

Application of Service-Oriented Modelling Method

This chapter presents a demonstration of how the new method was applied. It demonstrates how the modelling language and the modelling process were applied to motivate and justify a new service-oriented modelling method. The research results, of the research in form of application of a new method will be demonstrated thought three research topics: (1) Pragmatic-driven specification of information systems, (2) Service-oriented modelling foundation for information systems analysis and design (includes service-oriented constructs, analysis patterns and a method for integration between the static and dynamic aspects) and (3) Transition principles to implementation-specific design that show various studies of transition from a service-oriented model to an implementation-specific design. These three research topics are related to three levels of architectural framework for the service-oriented modelling process. The research results of these three topics contributed to the development of a systematic and integrated service-oriented modelling method, which has several new features. The main differences from other known modelling methods are the following:

• it provides a systematic modelling way through three levels of abstraction, which is necessary to bridge design requirements and technical design. It controls the fitness between business and information technology issues and can be used for improving communication between two types of stakeholders: business and design experts. This modelling method provides a modelling language, a modelling process and a way to transition from one level to another. It provides the guidelines and techniques for bridging between levels. The service-oriented modelling method provides a controlled way of modelling that helps to communicate the domain semantics unambiguously.

• the service-oriented modelling method enables the integration of static and dynamic aspects into a single modelling notation. Coherence of both aspects is critical in order to facilitate reasoning and to define the holistic understanding of enterprise architecture. Semantic integrity of the static and dynamic aspects can be achieved by combining external and internal views of services together. The interplay between intersubjective and objective perspectives, using a single service-oriented construct, contributes to this integration. Various static and dynamic dependencies provide the possibility to identify and eliminate such undesirable characteristics of system specifications as ambiguity, redundancy, incompleteness with respect to goals, and inconsistency. These criteria are critical for reaching a better quality of IS specifications.
• the method shows how to bridge computation-neutral models and implementation-specific design. Service-oriented diagrams redefine semantic details that can be used to elicit object-oriented diagrams. Service-oriented diagrams have no implementation bias and therefore can be used for bridging a communication gap among system designers and business experts.

Figure 22 shows the relationship between the architectural framework and the research topics.

![Figure 22: Presents an overview of architectural framework and research topics](image)

The first research topic, Pragmatic-Driven Specification of Information Systems, contributes to the systematic modelling method for structuring pragmatic knowledge about different services using pragmatic dependencies, and defines the ‘why’ aspect of the problem domain. The intentions of business experts are represented in terms of a set of pragmatic dependencies, which drive the overall system engineering process. Pragmatic specifications aim to provide justification for conceptual representations of enterprise components at the semantic level.

Research topic two, Service-Oriented Modelling Foundation for Information Systems Analysis and Design, contributes to the semantic integration of the static and dynamic aspects. The tacit structure of services, including structural and dynamic aspects, is important for systematic analysis of service architectures. It is crucial for visualization and to maintain the holistic representation of external and internal views of the same system. This topic also contributes with service-oriented constructs, modelling steps and analysis patterns. The third research topic, Transition Principles to Implementation-Specific Design, contributes to the knowledge that shows the transition from computation-neutral to implementation-specific design. This research contributes to the design of technical components. It is shown how service-oriented constructs can be mapped to technical components and object-oriented diagrams.
At the core of this thesis are a set of peer-reviewed papers, published in different conference proceedings (7) and books (4). The research results presented in the papers contributed to the ultimate goal of this thesis: to develop an integrated service-oriented modelling method for the analysis and design of information systems. Table 5 presents the connection between the publications and research topics.

**Table 5: The relation between research topics and publications**

<table>
<thead>
<tr>
<th>Research Topics</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5. Towards the Enterprise Engineering Approach for Information System Modelling across Organisational and Technical System Boundaries</td>
</tr>
<tr>
<td></td>
<td>6. Pragmatic-Driven Approach for Service-Oriented Analysis and Design</td>
</tr>
<tr>
<td></td>
<td>3. On Desirable Qualities of Information System Specifications</td>
</tr>
<tr>
<td></td>
<td>4. Enterprise Modelling Approach for Information System Engineering</td>
</tr>
<tr>
<td></td>
<td>7. Service-Oriented Foundation and Analysis patterns for Conceptual Modelling of Information Systems</td>
</tr>
<tr>
<td></td>
<td>10. Semantic Framework for Information Integration Using Service-Oriented Analysis and Design</td>
</tr>
<tr>
<td></td>
<td>11. How Models and Methods for Analysis and Design of Information Systems can be Improved to Better Support Communication and Learning</td>
</tr>
<tr>
<td>3. Transition Principles to Implementation-Specific Design</td>
<td>2. Extending Lyce Methodology using the Enterprise Modelling Approach</td>
</tr>
<tr>
<td></td>
<td>8. Introducing Service Orientation into System Analysis and Design</td>
</tr>
<tr>
<td></td>
<td>9. A New Method for Conceptual Modelling of Information Systems</td>
</tr>
</tbody>
</table>

The results of all research topics are presented in the next three sub-chapters.

### 5.1. Pragmatic-Driven Specification of Information Systems

This research topic presents a pragmatic-driven method (Gustas & Gustiené, 2002, 2004, 2008) for service-oriented information system analysis and design. It shows the modelling way and demonstrating example of pragmatic-driven specifications of IS and how these specifications are further analysed at a semantic level. Business solutions made by business experts can be represented in terms of a set of pragmatic dependencies, which drive the overall system engineering process. It contributes to the modelling way of refinement of pragmatic entities into service interactions.
Service architecture starts from the premise that every business process has a design (High et al., 2005). An ambition of service-orientation is to provide system designers with a constructive way of integrating businesses as a set of linked services. Pragmatic aspects are supposed to drive a system engineering process from business goals to service interactions. Models built at the semantic level are justified by pragmatic descriptions, which are defined in terms of goals, problems and opportunities. Pragmatic specifications determine a conceptual representation of service architectures at a semantic level. Service interactions provide justification for conceptual representations of static and dynamic aspects. Service-oriented analysis and design enables fitness between the pragmatic level, which defines the overall business strategy, and semantic level that defines business processes across organisational and technical system boundaries.

Pragmatic entities such as goals, problems, and opportunities can be related by pragmatic dependencies and analysed in different situations. Any business process functionality can be defined as a service. A service, in different contexts, from pragmatic point of view can be regarded as different pragmatic entities such as a problem, opportunity or a goal. Business activities can be defined in terms of a set of interaction loops between service requester and a service provider, linked to design goals (Gustas & Gustiené, 2008). Behind every business process is a clear motivation or goal, which can be analysed together with a final process state. The achievement of this state should bring value to customers. If a goal is not achieved, it could be possible that the business process goals were not correctly stated. This results in semantically incomplete business process specifications.

Pragmatic dependencies provide the business analysis with guidelines to deal with the incompleteness of the semantic descriptions of service architectures, since business processes are linked to design goals, and used to analyse pragmatic intentions of the actors involved in the business processes (Gustas & Gustiené, 2004). Pragmatic dependencies provide a hierarchical structure of goals. This structure is a basis for reasoning about the semantic completeness of system specifications. It should be noted that traditional modelling approaches have difficulties representing the scope and boundaries of the entire system (Milton, 2007). Pragmatic-driven specifications are useful, because they motivate service interactions and events. So they can be used as driving force in service-oriented modelling of service architectures. Pragmatic hierarchies and dependencies provide the motivation behind business solutions for the entire system development process.

5.1.1 Goals, Problems and Opportunities

Pragmatic dependencies can be viewed as modelling bases to reason about the intentions of designers related to the new system architecture. Goals, problems and opportunities are introduced using pragmatic links. An opportunity is a desirable fragment of an existing semantic specification that is intended to be maintained in a
new system. The desirable features of a problematic situation can be indicated as opportunities by using refinement links. A meta-model of an opportunity that is represented as a desirable organisational strength in a problematic situation is illustrated in figure 23.

![Figure 23: Opportunity as strength](image)

Opportunities represent fragments of the semantic specification that are intended for reuse in the desirable situation. Such fragments typically represent an organisational strength. Another type of opportunity can be defined as a situation that negatively influences at least one of the problems and positively influences at least one of the organisational goals. A meta-model of an opportunity that negatively influences a problem can be represented by a diagram, which is illustrated in figure 24.

![Figure 24: Meta-model of opportunity](image)

One way to improve an existing problematic situation is by eliminating the semantic fragment of specification that is represented as a problem at the pragmatic level. Another way for improving a current situation is weakening a problem by introducing a new opportunity, which has a negative effect on the problem. Such opportunities represent new features of a desired situation that should be maintained in the system reengineering process. The semantic specification of a problem should
be considered a part of a specification of the actual situation. On the other hand, opportunities and goals represent what is desirable in a new system. Goals can be used to specify the desirable changes that are represented at the semantic level in terms of interaction loops between actors.

Opportunities describe various possible alternatives for solving a specific problem and reaching a goal. If a specific opportunity is introduced then its semantic fragments must be integrated into the desired specification. An opportunity cannot be defined by only representing its negative influence to a problem. Problems are not completely clarified without stating a goal. A pre-defined goal can be achieved by avoiding a problem or using an opportunity for reaching a goal. Achievement of a goal for one enterprise actor can be regarded as a problem for another actor. Relativity of problems and opportunity is represented in the following figure 25.

![Figure 25: Relativity between problem and opportunity](image)

In this current situation, the company Travel Agency has the problem (p) 'Sharp Competition among Travel Agencies'. But for Customer this problem could be seen as an opportunity (o) in reaching his goal to get 'Attractive Prices of Trips'. Figure 26 presents another case of relativity between goal and problem.
In this example, Travel Agency has one of the goals to reach ‘High Profitability’ of provided services. For Customer, this goal could be viewed as a problem for reaching his goal to get ‘Attractive Prices of Trips’. High profitability for the company implies higher prices for Customer and it becomes a problem, which negatively effects customer’s goal ‘Attractive Prices of Trips’.

The two examples illustrate the relativity of interpretation of pragmatic entities as goals, problems and opportunities by different actors. Semantics of pragmatic entities depends greatly on the situation of the problem domain. An analysis of goals and problems helps business experts to explain their decisions and this idea can be clarified by using a hypothetical semantic difference operation, which is supposed to compute a semantic difference between two graphical descriptions. In conventional system development modelling approaches, the formalization of operation on pragmatic entities is problematic, because the static and dynamic aspects are analysed in isolation.

An overall goal can be defined as a semantic difference between a desired situation $S_D$ and an actual situation $S_A$. This could be represented as following: $S_G = S_D - S_A$. Here ‘-’ is the semantic difference operation. Any specific goal can be viewed as a further refinement of $S_G$. It denotes a conceptual description, which represents what is desirable in a new system (Gustas, 1997).

A problem can be considered as opposite to a goal (Gustas & Gustiené, 2008). A computation-neutral definition of a problem can be defined as a semantic difference between the actual situation $S_A$ and the desired situation $S_D$. Any specific problem describes what is not desirable in a new system and can be regarded as a further refinement of $S_P$. This could be represented as following: $S_P = S_A - S_D$.

The interpretation of problem and goal has some implications on how different pragmatic notions are perceived by stakeholders. In the initial phase of modelling,
the goals and problems can be analysed in isolation, but shared knowledge and consensus on a specific problem or goal cannot be reached without a detailed analysis of a set of semantic dependencies in the actual desired situation at the semantic level. The understanding of a stated problem cannot be reached without a complete understanding of what is desired in the new system. On the other hand, the semantics of a goal cannot be specified without understanding what is undesirable in a problematic situation. Sometimes, a problem makes sense even if the goals are not completely understood, but the reasons of problems cannot be fully comprehended unless they are refined in terms of the semantic dependencies in a problematic situation.

5.1.2. Operations at Pragmatic Level

Similarity of goals, problems and opportunities is an important characteristic since similar pragmatic entities can be placed in the same hierarchy and can be characterized by consistent interaction loops. Compatible interaction loops are defined in terms of:

- identical or similar actors involved.
- identical or similar pre-condition or post-condition object classes.

Similar concepts are connected either by the inheritance or the composition link. Similarity of pragmatic entities is useful, because it allows for performing two pragmatic operations, such as the operation of union and the operation of intersection. Since the interpretation of goal, problem and opportunity is relative and depends on the judgement of an actor; in this section we will sometimes refer to all of these pragmatic categories as goals. Any two similar goals, B and C, have their greater common specialization denoted by intersection (B \( \cap \) C) and their least common generalization denoted by union (B \( \cup \) C) (Gustas, 1997). The union of goals (u) is defined as the semantic union of their specifications. If any two goals, B and C, are a refinement of the goal A, then their union is the specialization of the same goal A as well.

Compositions of service descriptions that represent goals, problems or opportunities can be formed by using the operation intersection (\( \cap \)), which is defined as an intersection of the semantic specifications of goals. Intersection is represented by a common interaction loop. If any goal, A, is a refinement of two goals, B and C, then it is a refinement of the intersection of those goals. Goal A can be formed if and only if pragmatic entities B and C are similar and consistent. Intersection (B \( \cap \) C) can not be formed if pragmatic entities B and C are inconsistent. Inconsistency situation at the pragmatic level is characterized by conflicting or contradictory goals. If sub-goals of the same goal are conflicting, then a global goal is contradictory. Any contradictory goal has at least two sub-goals that are conflicting. Graphical notation of the pragmatic union is presented in figure 27(a), and the intersection is presented in figure 27(b).
The example of pragmatic union operation is illustrated in figure 28.

![Graphical representation of the union and intersection operations](image)

**Figure 27:** Graphical representation of the union and intersection operations

The union of sub-goals: *Sell a Trip* and *Rent a Car* can help to reach goal *Highly Competitive Services.*

Satisficing dependency $\circlearrowleft$ (Gustas, 1997; Gustas & Gudžienė, 2008) is another dependency used in the analysis of pragmatic entities. It is defined by the help of two refinement dependencies into opposite directions. Satisficing dependency means attaining a certain minimum quality in describing a goal that is sufficient to solve a problem. A satisfies B if and only if A is a refinement (i.e. B satisfy A) of B and a set of semantic dependencies of B is sufficient for the specification of a goal A. The notion of satisficing has been introduced by Simon (Simon, 1996). It is made up from two words: satisfy and suffice. This dependency is very important as it is used to bridge enterprise model fragments that are defined at the pragmatic level as goals, problems and opportunities.

Pragmatic decomposition dependency is another important dependency. The criterion of independence between pragmatic entities is important for identification of the decomposition links among goals. Pragmatic entities are independent if and only if they have no common interaction loops and they have neither negative, nor positive influence to each other. If any two goals A and B are independent, then their intersection empty ($\bot$), i.e. $(A \cap B) = \bot$. Here: $\bot$ is empty element. A is decomposition of elements B and C if and only if the components B and C are independent and their semantic union satisfies the goal A, i.e.

$(B \uplus C) \circlearrowleft A$ if and only if $(B \cup C) \circlearrowleft A$ and $(B \cap C) = \bot$.
Here \( \triangledown \) is the pragmatic decomposition operation. A graphical representation of the pragmatic decomposition is presented in figure 29.

![Graphical representation of pragmatic decomposition](image)

**Figure 29:** Pragmatic decomposition

An example of pragmatic decomposition operation is presented in figure 30.

![Graphical representation of pragmatic decomposition](image)

**Figure 30:** Presents pragmatic decomposition

Decomposition of the goal *Sell a Trip* in this case is complete. The satisficing dependency between sub-goals and goal means that achievement of two sub-goals is enough to reach the goal *Sell a Trip*. Sub-goals *Reservation of Trip* and *Paying for Trip* are independent and their intersection is empty. A goal decomposition hierarchy is usually formed of satisficing goals on the neighbouring levels of abstraction. Objectives at the bottom level are defined in terms of the basic semantic dependencies. Goal decomposition hierarchy describes how the various high-level objectives are going to be achieved.

Operations on goals create challenges for the traditional approaches, because the static and dynamic aspects are defined in different types of diagrams. In the presented service-oriented modelling method, semantic fragments of computation-neutral specifications are defined by using the service-oriented concept of an interaction loop. Interaction dependencies specify services in terms of communication actions, which play an integrating role in binding the static and dynamic aspects together. Operations with goals, problems and opportunities done at pragmatic level should have consequences on a semantic level.
5.1.3. Interplay Between Semantic and Pragmatic Representations

A Travel Agency modelling example is presented for the demonstration of the interplay between pragmatic and semantic representations. It is demonstrated how goals stated at the pragmatic level will be analysed and decomposed in terms of interaction loops and how these loops will be analysed at the semantic level. The example demonstrates how goals, problems and opportunities can help system analysts to reason about the advantages (positive influence dependency) and disadvantages (negative influence dependency) of business process activities.

Travel Agency has identified two problems in the current situation: *Sloppy Selling Process for Low Cost Trips* is an internal problem and *Sharp Competition among Travel Agencies* can be viewed as external problem. Both problems hinder the achievement of the goal *First Choice of Customers*, which characterizes the overall desired situation. It is the goal to attract and retain customers by finding the best ways to redesign the business processes in the Travel Agency. Two opportunities can be used to improve the current situation. The first opportunity *Offer Discounts for Returning Customers* represents system functionality, which is not available in the problematic situation. The second opportunity *Sell a Trip* characterizes functionality, which represents strength of the current situation. It is always desirable to maintain an existing strength for the future situation. If one of the goals can be reached, then the problematic situation will be improved. Interplay among the pragmatic dependencies for Travel Agency is graphically represented in figure 31.

![Figure 31: Pragmatic dependencies of Travel Agency](image)
Value creation through services is described in terms of linked related situations. With the help of a refinement link it is possible to decompose the main goal to see its structure in terms of a set of activities, and to understand how various actors contribute to the achievement of the main goal. Such an analysis is very important for the comprehension of the entire meaning of the goal. It is not sufficient to represent the specification of every sub-goal in terms of service interactions. Interactions must be justified by goals and integrated with behavioural aspects of conceptualizations.

The goal First Choice of Customers is refined into sub-goal Highly Competitive Services, which consists of three sub-goals Sell a Trip, Rent a Car and Offer Discounts for Returning Customers. In this example, two sub-goals, Sell a Trip and Offer Discounts for Returning Customers, will be analysed.

Opportunity ‘Sell a Trip’ represents a fragment of functionality that is viewed as a strength of the organisation, because it is a refinement of both the problem Sloppy Selling Process for Low Cost Trips and the goal Highly Competitive Services. It is also a strength because this activity was already used before in the organisation. This opportunity is decomposed into two activities: Reservation of Trip and Paying for Trip. It is the only alternative to Sell a Trip at the Travel Agency in the current situation, which is viewed as problematic. The goal decomposition process could be seen as a means of eliminating the problematic fragments of specification at the semantic level and complementing them by the desirable fragments. Decomposition hierarchy lies behind the strategies and the ways in which the high-level objectives are achieved. At the bottom level of abstraction, it is necessary to refine goals in terms of interaction loops between service requesters and service providers. This is represented in figure 32.
Sub-goal Reservation of Trip is decomposed into three activities: Searching for Trips, Selecting Trips, and Create Reservation. This decomposition is complete, because the semantic union of these three interaction loops is sufficient for Reservation of Trip. The goal decomposition hierarchy represents two actors, Customer and Travel Agent (service requester and service provider), that are involved in each business process step. The refinement of every goal at the bottom level is defined in terms of input and output flows that define atomic interactions in different service steps. Composition of interaction loops should result in an integrated business process or service. The whole service Reservation of Trip consists of the composition of three interaction loops that are represented in figure 33.
Figure 33: Description of sub-goal ‘Reservation of Trip’ at semantic level

In this example, the sub-goal Reservation of Trip was decomposed into three activities necessary to execute to fulfill the overall goal. These activities were defined using interaction loops between Customer (service requester) and Travel Agent (service provider, which is a computerized system). Customer as service requester initiates the first action Search Trip providing Trip Requirements as an input (information flow) for the action. Action Search Trip is viewed as a service request. Travel Agent is a service provider, which is supposed to activate the action Show Trip and provide Service requester (Customer) with Trip List. The interaction loop is complete if both actors
commit to their responsibilities. In the next interaction loop, a Customer selects a trip from a Trip List by initiating the communication action Select Trip. When a trip is selected, a Travel Agent shows registration form. In the third interaction loop, Customer is supposed to Enter Customer Data. If the customer does this, then Travel Agent will be obliged to Confirm Reservation.

New opportunities, which positively affect the goal of organisation, should be fully integrated into the overall business process. If a designer is willing to take advantage of an opportunity, then it is not sufficient to represent it on the diagram by using a refinement dependency. The enterprise model must unambiguously demonstrate how an opportunity is integrated at both the semantic and the pragmatic levels. For instance, the opportunity Offer Discounts for Returning Customers (see figure 31) embraces a new service, which is yet unavailable in the problematic situation. Therefore, a designer can take advantage of a goal decomposition hierarchy, which is introducing a new alternative to Sell a Trip by explicitly specifying what it means to Offer Discounts for Returning Customers. A more detailed refinement of the new opportunity is represented in figure 34.

Figure 34: Refinement of sub-goal ‘Offer Discounts for Returning Customers’

A new opportunity must be integrated into the existing enterprise model at the semantic level. It means that the Offer Discounts for Returning Customers functionality must be consistent with the other sub-goals in the hierarchy of Highly Competitive
Since both pragmatic entities, Offer Discounts for Returning Customers and Reservation of Trip, belong to the same hierarchy of Highly Competitive Services, they must satisfy conditions that apply for the greater common specialization Sell a Trip. Sell a Discount Trip is a new alternative of selling, which can be used by the returning customers. Discounts can be offered just to those customers, which have the possibility to log in. A new alternative for the achievement of goal Sell a Trip is explicitly introduced by extending the goal hierarchy. Two alternatives are represented in figure 35.

![Figure 35: Two alternatives of goal ‘Sell a Trip’](image_url)

The alternative way to Sell a Trip is represented by using a satisficing dependency link. Note that two sub-goals Sell a Trip for not Logged in Customer and Sell a Discount Trip can be interpreted as two different strategies (Rolland, 2005), which are expressed in terms of decomposition. This hierarchy tells a designer that the same Paying for Trip functionality is used in both alternatives. The semantics of Reservation of Trip differ significantly from Reservation of a Discount Trip. It consists of Offer Discounts for Returning Customers and Confirmation of Offered Trip. The sub-goal Paying for Trip represents an overlapping part, which can be represented as an intersection of two alternative goals (strategies) for the achievement Sell a Trip goal.

It is not difficult to see that a content of interaction loops is prescribed by the pragmatic part of the enterprise model. Goal hierarchies help a designer to control the semantic completeness of specifications or to identify the business process boundary. A discontinuity of an interaction loop is regarded as incompleteness in a business process description. Interaction loops have of paramount importance, because they define an elementary skeleton for a semantic description of a goal. Interaction loops help system designers to control integrity between the static and dynamic aspects. The transition and interaction dependencies must be integrated in the same business process (Gustas & Gustiene, 2004). Additionally, the semantics of business processes, which are defined by a greater common specialization in goal hierarchy, cannot be expressed in terms of contradictory goals. If new opportunities can be integrated with the existing business process descriptions, then they are
viewed as sub-goals. In this example, the opportunity *Offer Discounts for Returning Customers* is viewed as a sub-goal for a *Trip Agency*. Thus, it cannot be contradictory in the *Sell a Trip* goal hierarchy. All semantic integrity problems must be resolved in the context of *Highly Competitive Services*.

The service-oriented way of modelling, based on interaction loops, provides not only the possibility to follow the intersubjective perspective, which defines the relationships between actors and their responsibilities, but also the picture of an objective perspective, which defines object changes taking place during certain activities. These changes can be traced by checking the changes in attributes that define the properties of the objects and explain the actions that take place during the interactions. If there are no changes in attribute values, then the usefulness of certain specific actions should be questioned and checked. This way of modelling using one modelling notation, provides a way to follow and trace the entire business process. Goal hierarchies provide a basis for reasoning about semantic integrity and semantic completeness control in system specifications. Pragmatic specifications done at the pragmatic level are business-oriented and they are useful for several reasons: they motivate service events and are a driving force for service-oriented engineering of business processes.

### 5.2. Service-Oriented Modelling Foundation for Information Systems Analysis and Design

This section presents the second research topic. It presents the foundations of a new service-oriented modelling method for analysis and design of information systems. It also provides a philosophical view and the basic constructs for service-oriented modelling. It demonstrates how these constructs can be used for the creation of system analysis patterns such as: sequence, synchronisation, iteration, selection and search. It demonstrates how various combinations of static and dynamic dependencies are able to express the main workflow control patterns. This is a unique feature of service-oriented modelling method.

The purpose of this new service-oriented modelling method is to provide a graphical modelling language notation and a set of design principles that enables enterprise architectures to be analysed in terms of services. The analysis is made on a high level of modularity, defining services as business units (Armi-Bloch et al., 2009). The foundation of this method is rooted in the ontological principles of the concept of service. It is based on the premise that the concept of service can be applied for the conceptualization (Gruber, 1995; Guarino et al., 2009) of enterprise system architecture in the form of organisational and technical system components. This architecture can be defined as a set of loosely-coupled components that interact with each other.

It is also based on the ontological features of the enterprise system, which is the composition of objects and interacting components (Bunge, 1979) that are
interrelated and have unity and integrity. According to Bunge (1977) objects can interact, and this interaction causes state changes to other objects. The main ontological principles taken for the foundation of the method are the following:

- the world is composed of things and everything changes. Things have properties. We study them and modify them to understand the changes that are the results of certain actions (objective perspective),
- things are grouped into systems. There are no independent subsystems. Every system interacts with other systems (intersubjective perspective).

A starting point of the ontological definition of an enterprise system in the presented service-oriented foundation is quite similar to the ontological understanding of system and enterprise as a system and enterprise ontology (Dietz, 2001, 2006b). Every enterprise can be viewed as a composition of organisational and technical components. The presented service-oriented method is based on the assumption that business process models are composed of loosely-coupled components which are viewed as service requesters and service providers. When interacting, they cause changes in the objects, which are defined by properties. Organisational and technical actors can be seen as loosely-coupled components, which can be decomposed using different dependencies. To capture the holistic structure of the problem domain, it is necessary to understand the static and dynamic aspects of enterprise components, and how different components are interrelated. In the enterprise, the interactions are built around long-term relationships, which reflect the life of the enterprise (Chesbrough & Spohrer, 2006).

The concept of service-orientation is used in different contexts and with different purposes, but there is one aspect that is constant in its existence and that is, that it represents a distinct approach for separating concerns (Erl, 2005; Greer, 2009; Jacobson & Ng, 2005). It means that the organisational and technical solutions of problems can be constructed and managed, decomposing them into smaller subsystems. Decomposition and refinement of services at a pragmatic level and analysing them using pragmatic dependencies help business analysts to see and understand the structure of goals (Gustas & Gustienė, 2008). By focusing on service architecture, in terms of how business people understand it, it is possible to make service-oriented analysis and design work more profitably. If the technology thinking occurs prior to business thinking, then the systems usually fail to deliver business value. Such modelling should be driven from the top down, directly from business requirements.

Service architecture is based on the assumption that business process models can be composed of loosely-coupled components (Erl, 2005; Krafzig et al., 2005), which are viewed as service requestors and service providers. According to the philosophy of the service-oriented foundation, every enterprise can be seen as a composition of organisational and technical components, which are viewed as various types of enterprise actors. The notion of a subsystem is fundamental in the presented service-oriented method, as every component can be seen as a subsystem. Analysis based on actors as components and subsystems helps to introduce the principle of separation of concerns, which is important for achieving independency of requirements.
Any two actors, either organisational or technical, can be linked by different semantic dependencies. The principles of the service-oriented method are based on designing interactions between different enterprise actors. Two interaction dependencies in opposite directions between actors form an interaction loop. Discontinuity of service interactions causes breakdowns in business process specifications that lower the quality of system specifications. Breakdowns indicate incompleteness of service interactions.

A new way of modelling provides improved integrity and traceability of service architectures by using comprehensible graphical descriptions across different perspectives and along different dimensions. The computation-neutral modelling of business processes in terms of services gives the possibility to define service-oriented analysis and design as an engineering discipline, which provides a basis for semantic integrity and semantic completeness control. It provides the possibility to identify undesirable semantic characteristics of system specifications (Gustiené, 2003b),(Gustas & Gustiené, 2004), which are difficult to achieve with contemporary modelling approaches. Adopting a new service-oriented paradigm has the potential to manage the complexity of information systems development, which contributes to a reduction of semantic problems of communication.

### 5.2.1 Core Elements and Perspectives of Service-Oriented Modelling

An analysis of the system means the creation of a shared set of concepts; to communicate these concepts during the development process, they should be unambiguous and comprehensible to all stakeholders involved. A domain's interrelationships constitute a large part of its implicit structure. A deep understanding of the domain relies on knowing how all objects of the problem domain are interconnected (Yoo et al., 2004). Very often ambiguous or incomplete requirements are the results of insufficient data analysis at the analysis level, and not being able to identify a complete set of relationships among different domain elements. This results in that incomplete IS specifications are passed on to the design phase, which causes inaccurate communication between the analysis and design phases.

Service-oriented analysis is based on modelling intersubjective and objective perspectives of an enterprise system across organisational and technical system boundaries. An intersubjective perspective defines how enterprise actors, which can be organisational or technical components, are related to each other. These actors can be viewed as service requesters and service providers. This perspective signifies certain commitments and responsibilities between enterprise actors, which are motivated at the pragmatic level (Gustas & Gustiené, 2008). The objective perspective defines how different objects change when the actions during the interaction process between actors take place. State changes of the objects define their behaviour. The objective perspective indicates the internal changes of objects. Actions cause transitions, which make changes in the attributes of objects. From an
objective perspective of the system, objects, actions and states are the three most important building blocks of the system. The ability to combine these two perspectives into a single and unified modelling notation provides the possibility to integrate static and dynamic aspects of service architectures. Such a modelling notation enables system designers to model structure and behaviour at the same time. Intersubjective and objective perspectives (Dietz, 2006b; Gustas & Gustiené, 2009b) lie in the foundation of the main construct for service-oriented modelling.

One of the main elements of service-oriented foundation is enterprise actors. Actors are subsystems that are represented by individuals, organisations and their divisions or roles, which denote groups of people. Technical actors are subsystems such as machines, software and hardware components, etc. Any two actors can be linked by inheritance, composition, classification or interaction dependencies, which are represented graphically in figure 36.

**Figure 36: Actor dependencies**

Inheritance dependency between actors is used for sharing the static and dynamic similarities. More specific actors inherit the composition and interaction dependencies from more general actors. Dependencies represent additional intrinsic actor interaction features and structural properties that are prescribed by an enterprise system.

Composition is a conceptual dependency used to relate a whole to other concepts that are viewed as parts. It is a stricter semantic relation as compared to an aggregation and a composition that is defined in the object-oriented approaches. Composition is characterised by the following properties: (a) a part cannot simultaneously belong to more than one whole of the same concept, (b) if it does belong to more then one whole, then it must be a whole that is an instance of another concept, (c) a part and a whole are created at the same time. Once a part is created, it is terminated at the same time the whole is terminated.

A classification link between two actors is used to define their instances. In conceptual modelling, an instance can be viewed as an element of a set that is defined by the concept it belongs to. In the same way as an object can be manipulated by operations, an actor has interaction privileges and responsibilities that are defined by the interaction dependencies.

Interaction dependencies are used to conceptualize services between various enterprise system actors. Since actors are implemented as organisational and technical system components, they can use each other, according to prescribed patterns, to achieve their goals. Two interaction dependencies in opposite directions
between a service requester and service provider define a typical action workflow loop (Denning & Medina-Mora, 1995; Gustas & Gustiené, 2009b).

Service providers are actors that typically receive service requests, over which they have no direct control, and initiate service responses that are sent to service requesters. A system can be defined as a set of loosely-coupled interacting components, which are able to perform specific services on request. Examples of actor dependencies are represented in Figure 37.

![Figure 37: Example of semantic dependencies between actors](image)

Service requests and service responses are understood as communication actions (Dietz 2001). We use the concept communication action in a general sense to describe cooperation action or an act of human behaviour with some intention or with some subjective meaning attached. In the service-oriented modelling method the distinction between communication and interaction is made. Communication takes place just between humans; interaction is between human-technical component, and technical-technical components. A communication/interaction link between two actors indicates that one actor depends on another by a specific action. It represents the intersubjective perspective of the communication/interaction action.

### 5.2.2. Intersubjective Perspective

The intersubjective perspective of an action indicates that one actor depends on another actor. This perspective emphasises the tasks of the actors and organisational units who carry out the process. It is represented by the interaction dependency (→) notation. The meaning of the interaction arrow is similar to a strategic dependency link that was introduced by the i* approach (Yu and Mylopoulos 1994), where actors are related to each other to achieve goals, perform tasks and furnish resources. The intersubjective perspective is typically distinguished by identifying a physical, information or decision flow between the two actors involved. This perspective is important, as it presents the actors as independent entities, who add value by performing activities (Gordijn et al., 2000). In the presented service-oriented modelling method, strategic dependency is considered at the same time to be an
action and interaction flow. Actions are carried out by actors, who are called agents. These agents can be seen as service requesters or service providers. Interaction dependency between these actors can be viewed as a communication channel for transferring an information, physical or decision flow from agent to recipient. Typically, an agent is sending a flow to a recipient in order to achieve his goal (Gustas and Gustiene 2004). The achievement of the goal will depend on a service provider, which should deliver a service flow into the opposite direction. The interaction loop between service requester and service provider should be a complete and closed loop in order to fulfil the goal.

Flows are concepts that are represented by rectangles. Solid boxes are used to represent physical flows and light boxes represent information flows. A communication action without an information or physical flow component specifies a decision. Graphical notations of three different flows are depicted in figure 38.

**Figure 38: Notation of information, physical and decision flows**

The intersubjective perspective helps to visualize who uses the data and to keep track of the data used. Traceability of data is important for data quality issues It can indicate discontinuity or broken business processes (Kimball, 2009), which is necessary to detect early, as it can cause discontinuity problems in later development phases.

### 5.2.3. Objective Perspective

The dynamic aspect of service includes not just interaction between actors as active elements but also the behaviour of passive objects that take place when service requests and service responses are evoked. The state changes of passive objects that are results of interactions define the internal behaviour and represent the objective perspective of the service. Objective perspective of an interaction loop represents static aspects. The composition of different types of communication actions results in diagrams that define a continuous or finite lifecycle for one or more objects. It is not enough to solely represent interaction between actors; it is important to define what semantic differences take place in the objects when the action is terminated. As
every action is motivated by a goal, it must result in some valuable changes for actors involved in the interaction loop. Semantic differences that take place in objects during the actions reflect the dynamics of the system. The objective perspective typically represents creation, termination or state change effects (Hull et al., 2003) in an enterprise system. Without the ability to represent noteworthy changes, designers would have difficulties understanding the rationale and effect of every action. If effects are not identifiable, then the action can not be considered as purposeful. The objective perspective concerns the representation of objects from a problem domain. From an objective stand point, a request and response action changes business data from one consistent state to another. Enterprise models should be able to clearly define the semantic details about the state of attribute values when the action takes place (Gustiené et al., 2009).

Services are dynamic subsystems, because outputs depend not only on inputs, but on service states as well. The dynamic aspect of a service can be characterised by using pre-condition and post-condition states. States define constraints on service objects and restrict service responses to the present and future inputs. Requests, responses and states are crucial to understanding the semantic aspects of services. Pre-condition states are important for determining a service output flow and a post-condition state.

According to Bunge (1977), all objects have properties (attributes) that characterise them. State changes that take place with objects when interaction is invoked define the objective perspective of an interaction loop. Object world changes are specified for every communication action by using a transition dependency link (→). The link represents internal changes of objects. Transitions are triggered by actions, which specify possible changes to occur in different objects. Any concept to be viewed as a class must be provided by the attribute dependency links. Attributes of the pre-condition object class in any action define types of instances to be removed, and attributes of the post-condition object class represent types of instances to be created. This is illustrated by figure 39.

Figure 39: Representation of the semantic difference between two classes

The properties of objects are defined by the mandatory attribute dependencies. They define an important semantic difference between two concepts. Lack of the noteworthy difference between a pre-condition and post-condition class indicates
that specification is incomplete or the communication action is not purposeful. Pre-
condition and post-condition classes in every action specify the permissible ways in
which changes may occur. In the conventional information system modelling
approaches, changes are represented by using state-transition links in a finite state
machine (Harel, 1987).

The attribute dependencies stem from the traditional data models. The semantics
of static dependencies in object-oriented approaches are defined by multiplicities.
They represent a minimum and maximum number of objects in one class that can be
associated to objects in another class. In the service-oriented modelling method, only
mandatory constraints, static associations without semantic holes, are used. A
graphical notation of the attribute dependencies and their cardinalities is represented
in figure 40.

![Graphical notation of the attribute dependencies](image)

**Figure 40:** Graphical notation of the attribute dependencies

This graphical notation corresponds to a classical way for representing associations
(Hoffer et al., 2004) between two entities. One significant difference of the notation
used in the service-oriented approach is that the association ends are nameless.
Dependencies are never used to represent relationships or mappings between two
sets of objects in two opposite directions. Whether a concept is regarded as a class,
attribute or relationship depends on types of the static and dynamic dependencies
through which they are related to other concepts. Additionally, any two concepts (in
the same way as any two actors) can be linked by inheritance, composition,
classification or interaction dependencies. States can be defined for a class or an
actor as well. Interaction dependencies between classes are used to represent static
relationships. Graphical notations of the remaining static dependencies are
represented in figure 41.
Similar attributes are inherited by more specific classes according to the inheritance link (→). The inheritance arrow denotes a specialisation and generalisation. It is always pointing to a more general concept. It is possible to distinguish between exclusive or overlapping as well as total and partial inheritance situations (Hoffer et al., 2004). All these cases can be expressed by using the exclusive specialisation and mutual inheritance link. Mutual inheritance dependency (↔) can be used for representing classes that are viewed as synonyms. It is defined as follows: A ↔ B if and only if A → B and B → A.

The composition of classes can be defined in the same way as the composition of actors. The presented modelling notation distinguishes among the single-valued and multi-valued composition. In general, the whole and a part are required to be created and terminated at the same time. In a multi-valued case of composition, a new part can be created for an already existing whole and sometimes a part can be terminated without removing a whole. Nevertheless, if the last part is terminated, then the whole is terminated at the same time. A part cannot simultaneously belong to more than one whole of the aggregate concept. If it does belong to more than one whole, then it must be a whole that is instantiated by another class. A transition action (including creation and termination) on the whole propagates to its parts and vice versa. Interaction dependency between loosely-coupled actors propagates to composite actors. Independently created are concepts whose objects are not composed of each other.

Intersubjective and objective perspectives were distinguished to conceptualise the organisational as well as technical parts of information systems. The objective perspective can be applied to represent the internal behaviour of the objects. It represents data, which is analysed in the context of interaction between organisational and technical system components. Interactions among different actors can be used to manifest object property changes that are important to elicit the semantic meaning of the problem domain. Figure 42 represents two perspectives of an interaction loop.
Integration of internal and external behaviour, which is encapsulated in a service concept, provides the possibility to express dynamic aspects that include interaction between external actors (intersubjective perspective), and behaviour that defines the states of objects (objective perspective). It also expresses structural aspects that define internal data. Such way of service modelling enables the integration of process and data. The cohesion of these perspectives into one new abstraction enables the representation of static and dynamic aspects using a single modelling notation. An advantage of a single modelling notation is that it provides semantic consistency control of the static and dynamic aspects. Semantic integrity of the static and dynamic dependencies in various interaction loops is one of the benefits of service-oriented analysis and design. Traditional information system modelling notations do not combine the static and dynamic aspects, representing them instead in totally different types of diagrams.

**5.2.4. Basic Semantic Constructs of Service-Oriented Modelling**

Basic constructs are fundamental semantic constructs for the service-oriented modelling method presented in the thesis. These constructs are based on two basic events, which are the main events used for service representations. Service representations are built by conceptualizing interactions among organisational and technical components, which can be viewed as various types of enterprise actors.

The purpose of the presented semantic constructs is that they can be used for service-oriented analysis and semantic integration of different modelling dimensions (Gustas & Gustiené, 2009b). There are two basic events fundamental to semantic service-oriented constructs: creation and termination events. They are basic for the
definition of a reclassification even that can be understood as a communication action (Dietz, 2001). Fundamentally, three kinds of changes are possible during any transition: an action is either terminating or creating an object, or it can perform termination and creation at the same time. A creation is denoted by an outgoing transition arrow to a post-condition class. A graphical notation of the creation action is represented in figure 43.

![Figure 43: Graphical representation of the creation action](image)

A termination action is represented by a transition dependency directed from a pre-condition object class. Before an object is terminated, it must be created. Since a future state makes no sense for a termination event, it is not included in a specification of action. Pre-condition class in a termination action can be understood as final in an object’s life time. The graphical notation of a termination action is represented in figure 44.

![Figure 44: Graphical representation of the termination action](image)

Reclassification of an object can be defined in terms of a communication action that is terminating an object in one class and creating it at the same time in another class. Sometimes, objects pass several classes, and then they are removed. A graphical notation of the reclassification action is presented in figure 45.

![Figure 45: Graphical representation of the reclassification action](image)
Pre-condition and post-condition classes typically define constraints on objects, which restrict the sending and receiving of communication flows between technical or business components. A reclassification action in a computerized system can be implemented either as a sequence of one or more object creation and termination (or read and update) operations. Request and response flows, together with created and terminated object classes, are crucial to understand the semantic aspects of services. A pre-condition object state and the input flow should be sufficient for determining a service output flow and a post-condition object state.

Service architecture can be composed of various interaction loops. The semantics of such composition can be defined using two or more constructs of basic actions. The composition of these three types of basic constructs is used for the conceptualisation of a continuous or finite lifecycle for one or more objects in the service loop. An example of the interplay of three basic events in one service loop is presented in figure 46.

**Figure 46: Illustrates basic service loop**

In this example, a flight reservation service is composed of a creation, termination and reclassification actions. The Request Flight action can be triggered by a Customer entitled to send Flight Reservation Data to a Travel Agent. Reservation Request may be viewed as a stored record in a web based ticket reservation system. If stored data satisfies a specific condition, then the Create Reservation action is invoked by a Travel Agent, which can be implemented as a software component. According to the presented diagram, the reclassification action will create a Reservation object and at the same time the object Reservation Request must be removed. Customer will be informed about his reservation by sending him Flight Reservation flow information.

The semantics of object changes is expressed using three types of actions: creation, termination and reclassification. A lifecycle of an object is typically represented by an initial, intermediate and final class. A creation event corresponds to a starting point and removal action – to the end point in an object’s lifecycle. The most critical issue in the modelling of the interaction details is the semantic integrity of the static and dynamic aspects. It is not sufficient to represent what type of objects are created and
terminated. For instance, *Reservation Request* and *Flight Reservation Data* in the example may correspond to a database view or they may be implemented as independent classes of objects. Service-oriented models and rules must clearly represent the semantic details of attribute values that must be either removed or preserved in any creation, termination and reclassification action. This is important for integrity constraints (IC), consistency (Arni-Bloch et al., 2009) checking and validation.

### 5.2.5. Main Steps of Service-Oriented Analysis

There are seven main steps that are fundamental to service-oriented analysis. Service-oriented analysis done using these steps supports the step-by-step principle, which is one of the principles of information system development. The justification of such an analysis is that step analysis supports a systematic and incremental way of modelling. Modelling steps are necessary to identify intersubjective and objective perspectives of service architecture. This identification is important for achieving the integration of interactive, behavioural and structural modelling dimensions. The identification of interaction loops and actions represents an intersubjective perspective that is the starting point of service-oriented analysis. Objective perspective represents data or a static part of a problem domain. This perspective is defined by identifying transition and attributes dependencies. It is important for the representation of the behavioural aspect. Without the ability to represent noteworthy changes in the objective part, it is difficult to understand the effects of actions. The ability to gradually identify these perspectives provides the possibility to control the modelling process and integrate the static and dynamic aspects of the problem domain.

Conceptual representation of service architecture can be defined using one or more interaction loops between enterprise actors that can be viewed as service requesters and service providers. The core elements of the service structure are actors (service requesters and service providers) and the interactions between them, flows (information, material, and decision), actions, transition, classes and attributes. The structure of a service can be modelled by seven main steps, which allow for keeping and following consistency between elements of service architecture. All steps are demonstrated using the Employment Service example. These steps are as following:

1. **Identification of actors and flows.** This is presented in figure 47.

![Figure 47: Illustrates actors and flows](image-url)
The first step implies the identification of actors, which according to the service oriented modelling method, can be seen as different components. This step defines the interaction of system components. A system can be defined as a set of loosely-coupled interacting components, which are able to perform a specific service on request. Actors are active concepts, which represent enterprise subsystems, the instances of which can be organisational or technical components. These actors are characterised by interaction dependencies and flows that are types of concepts representing information flows or material things. A communication action without any concept flow implies decision or control.

2. Identification of static dependencies among actors. This step identifies static dependencies among the actors involved in an interaction process. It is important for a good semantic analysis of concept (actors) structure (Gustiené, 2003b). Actors represent a set of objects from the physical world. They are active concepts representing dynamic sub-systems. During the second step, the semantic dependencies of these actors are defined. Any two actors can be linked by inheritance, composition and classification dependencies (Gustas & Gustienė, 2009b). Actor dependency links provide the possibility to describe the invisible side of enterprise architecture. Actors are the active elements of an enterprise, the ones who initiate the actions. The active life of the actors depends greatly on how their actions are guided and why they act. This step is presented in figure 48.

![Figure 48: Illustrates static dependencies among actors](image)

The conceptualization of active actors and their dependencies represent the intersubjective perspective of the service-oriented modelling method. This step allows for the comprehension of the system as a composition of interacting components.

3. Identification of actions. This step is about adding actions to flows. Actions carried out by actors during interaction can be seen as a communication channel for transferring an information or physical flow from one actor to
another. It is presented in figure 49.

Figure 49: Illustrates identification of action

The identification of actions is important, since they provide the basis for the objective perspective of the system. Actions motivate and prescribe the changes of objects. Every action is goal-driven, which means that certain changes should take place. The actions that do not change anything are noteworthy.

4. Identification of transitions, pre-condition and post-condition classes (passive concepts). This step defines the behavioural part of the system. This step is presented in figure 50.

Figure 50: Illustrates transitions, pre-condition and post-condition classes

During this step, the changes for every communication/interaction action are specified by identifying transition dependencies between concepts. Internal changes of the objects are defined using transition links between concepts. Transitions are
usually triggered by actions, which enable changes in different objects. During the action, a new object is created or terminated in pre-condition and post-condition classes. If a termination and creation action is performed at the same time, then it is called a reclassification action.

5. **Identification of class attributes.** This step is important for the identification of semantic differences that take place during object transitions in pre-condition and post-condition classes. It defines the differences in the static part of the system. These differences are identified by attribute dependencies. These dependencies are useful for the definition of semantic differences or similarities of the concepts, because the semantics of the concept can be stated depending on the types of dependencies that relate it to other concepts. If specified or derived attribute dependencies are contradictory, then the diagrams are inconsistent. Attribute dependencies are important for the identification of semantic redundancy. If inferred dependencies were represented in the same diagram, then they would indicate a situation of semantic redundancy (Gustiené, 2003b), which negatively affects the semantic quality of IS specifications. The identification of class attributes is presented in figure 51.

![Figure 51: Illustrates class attributes](image)

Noteworthy changes can be understood following the service-oriented diagram. The Employ action terminates the object of Applicant and creates the object of Employee. When Applicant was reclassified into Employee, the invoked operation Employ deleted the object Applicant and Application with the attribute Reference Number and created the object Employee with two specific attributes Employment and Position.
6. *Adding of alternative actions.* This step is important, because the modelling process should support not just the core business process but provide the possibility to show possible ways of actions. Contingent processes can be demonstrated using alternative actions. One of the advantages of service-oriented modelling is that it allows for showing the alternative and the core course of events in the same diagram. The core course of actions and an alternative action is represented in figure 52.

![Diagram](image)

**Figure 52:** Illustrates alternative actions

When the applicant applies for a job, the organisation can have two alternatives: to employ him or reject his application. If his application is rejected, application (flow *Application*) will be sent to him. If he will be employed, he gets the information about it (flow *Employment Data*), and the object of *Applicant* will be reclassified into *Employee*. Both cases are explicitly presented using a single diagram.

7. The seventh step represents the *refactoring process.* This process is essential to keep the design clean from inconsistencies and redundancies and to keep it understandable. This process does not alter the external functionality but makes the necessary changes to the internal structure. Classes and their attributes must be revisited and their semantics examined several times. It could be necessary for composition or generalization issues. In the example, presented in figure 53, the bottom-up generalization (Blaha & Rumbaugh, 2005) is done.
It is the process of searching and checking for classes with similar attributes, and slightly redefining some attributes or classes to fit it. Super-class Person generalizes two classes: Applicant and Employee. The inheritance mechanism allows for sharing attributes via a generalisation/specialisation relationship. Applicant and Employee share the attributes Name and SS Number, and have their own specific attributes. Sub-class Applicant has its own specific attribute Application and sub-class Employee has specific attributes such as Employment and Position. The refactoring process is essential, because it makes necessary structural changes in order to make conceptual models clean from inconsistent and redundant attributes, otherwise the diagrams will grow in size.

Such an incremental way of modelling, by adding new semantic information at every step, provides a systematic and integrated way to manage the complexity of a problem domain as well as the complexity of diagrams. Such way of modelling resembles the way the buildings are built, from components, by adding necessary information, until the final product is finished. The advantage of such way of modelling is that it provides not just the framework of what should be analysed but also how it should be done, providing the guidelines for the modelling process. The method also provides the rules and ways to identify and check the undesirable system characteristics (Gustas & Gustiené, 2004) that ensure a better semantic quality of IS specifications. The refinement of service interactions during the modelling process helps to reduce semantic misunderstandings (Gustiené, 2003b) among stakeholders. Such a way of modelling, designing graphical representations, can be used for the validation and verification of system requirements.
5.2.6. Analysis Patterns for Service-Oriented Analysis and Design

Basic service-oriented constructs can be used for constructing system analysis patterns. They are similar to workflow patterns (Russell et al., 2006) which were established with the purpose of delineating the requirements that arise during business process modelling on a recurring basis and to present them in a semi-formal description. One of the main contributions of this thesis is the presentation of such patterns. It was demonstrated that the service-oriented modelling language is sufficient for defining the main system analysis and design patterns such as sequence, synchronisation, iteration, selection and search. Workflow patterns are usually defined by using Business Process Modelling Notation (BPMN) (2004) for Business Process Diagram and UML Activity Diagram from the Object Management Group (OMG) (2009). Both notations are able to express process behaviour but do not take into account the static part of the business process. It does not explicitly show what happens with the objects when an activity takes place.

Five main analysis patterns for service-oriented analysis and design are presented in this thesis. The examples of corresponding behaviour are presented as well. Contrary to traditional workflow patterns (White, 2004), the analysis patterns of the business process presented in the thesis are constructed by combining static and dynamic dependencies. Service-oriented constructs used to represent the patterns are defined as an interaction loop between service requester and service provider. The expressive power of static and dynamic dependencies is sufficient for defining main workflow patterns, such as: sequence, selection (choice and merge), synchronisation (split and join) and iteration (Gustas & Gustièn, 2009b). The semantics of service architecture can be defined by using one or a combination of more interaction loops. Each interaction loop is composed of creation, termination or reclassification actions. By matching the interaction dependencies from agents to recipients, one can explore opportunities that are available to actors. The static dependencies define complementary semantic details of interactions, which are important for reasoning about service architecture patterns.

Modelling patterns for service-oriented analysis and design are important for two major reasons. Firstly, they can be used for demonstrating the interplay of fundamental constructs that are used in system analysis and design process. Secondly, patterns are important for the evaluation of the expressive power of semantic modelling languages (Rad et al., 2009). Comprehension and visual recognition of the fundamental patterns is necessary for building more specific pattern variations by composing them in different ways.
5.2.6.1. Service Interaction Pattern: Sequence

The *sequence pattern* is defined by an ordered series of activities. One activity starts after a previous activity has completed. The sequence pattern can be defined by using a composition of two or more reclassification actions. Since a creation and termination action is a special case of reclassification, it can be used instead of a reclassification action. An example of a sequence pattern is represented in figure 54.

![Figure 54: Sequence pattern](image)

The example of this sequence pattern is presented in figure 55.

![Figure 55: Example of sequence pattern](image)
In the presented example, the sequence of three creation actions is used to express the sequence pattern: Send Bill, Pay, and Confirm Payment. Pay action can be executed only if Send Bill action has been completed and Confirm Payment action can be processes only if the process of payment has been completed.

The changes that take place with the objects present the static aspects of the system (objective perspective). When Travel Agency sends Bill to Customer, Send Bill action creates a new object Reservation in state [Bill Sent], which is a specialization of object Reservation, which was created in previous interaction loop. The object has two specific attributes: Reservation[Bill Sent].Data and Bill. When Customer gets bill and evokes the process of payment, action Pay creates a new object Reservation[Paid] with a specific attributes Payment and Reservation[Paid].Data.

Every action is responsible for the removal of the attribute object links that are associated with the pre-condition class and for the creation of attribute object links with post-condition class. It cannot be responsible for any changes of object links of more general classes or for object links of the attributes. Creation, reclassification and removal of objects in more general classes, and in the attributes that are viewed as classes with their own attributes, should occur in an earlier sequence. For instance, for the Send Bill action to be triggered, a Reservation object is required to be created in advance by another service.

5.2.6.2. Service Interaction Pattern: Synchronisation

Sometimes, some activities must be performed concurrently rather than serially. A Synchronisation pattern combines the path of these activities. It is important that the final set of activities be completed before the next process can continue. The synchronisation pattern is presented in figure 56.

![Synchronisation pattern](image)

**Figure 56: Synchronisation pattern**

This pattern illustrates that an action is responsible for the removal of object A and all its parts B. Creation of D requires the creation of at least one object of E. Compositional attribute objects must be created, reclassified or terminated by the same action, because a part and a whole have identical life cycles. If an object is created, then the links with the compositional part are created as well. If an object is
deleted, then the links are deleted/disconnected at the same time. That is the reason why an action propagates according to the class composition links from a whole to a part and vice versa. Propagation of actions is a useful property, because it allows modelling synchronisation in a natural way. The synchronisation pattern example is illustrated in figure 57.

![Figure 57: Example of synchronisation pattern](image)

Supply action propagates to parts: termination of Order and Order Items, and creation of Delivery and Delivery Items. Food (material flow) means the delivery of a set of items, which were Order Items.

5.2.6.3. Service Interaction Pattern: Iteration

Iteration pattern is a special case of sequence, where a post-condition class of the response action plays role of a pre-condition class for the service request. The pattern is represented graphically in figure 58.

![Figure 58: Iteration pattern](image)
The example of iteration pattern is demonstrated with an interaction loop between two actors: Customer and Book Store (can be understood as a computerized system), which is presented in figure 59.

![Figure 59: Example of iteration pattern](image)

If Customer wants to buy some books, he enters the quantity and triggers the action **Enter Quantity**. The action creates the object Product in Order [Modified], which has one entered quantity number (see the attribute Modified Quantity). When the book store receives the message, it invokes the action **Show Modified Quantity** and sends the information flow of Modified Quantity to the Customer. The Modified Quantity attribute is consumed by Modified Quantity flow in this action. The action **Show Modified Quantity** terminates the object Product in Order [Modified] and creates the object Product in Order, which will be input to the action **Enter Quantity** in a new service interaction.

**5.2.6.4. Service Interaction Pattern: Selection**

A Selection pattern can be expressed using a composition of two different sequences between the same two actors. The selection represents two alternative outcomes of a service request that can be selected by service provider. Two ways of replaying by service provider are mutually exclusive. Typically, only one type of response is desirable by requester. The selection pattern is represented graphically in figure 60.
Note that Response 1 and Response 2 are exclusive. If Response 2 is triggered, then pre-condition class object B is removed and response 1 cannot be triggered and vice versa. The example of selection pattern is represented in figure 61.

For instance, if Person (service requester) applies (Apply) for a job by sending application (Application Data flow) to Company (service provider), then Company has two alternatives to choose between: to employ a Person or reject an Application. The Apply action should create an Application object, which is associated to one Applicant. Every Applicant object can be composed of one or many applications. In the case of an Employ action, Applicant object should be reclassified to Employee. Otherwise, the Reject action should terminate Application object.
5.2.6.5. Service Interaction Pattern: Search

A search pattern can be defined using a composition of sequence and iteration patterns. An iteration pattern with a missing pre-condition class for Request and post-condition class for response would express a special case of a search pattern, where a found object (B) is created and then consumed in a response action, which is presented for requester. A search pattern is represented graphically in figure 62.

![Figure 62: Search pattern](image)

A post-condition of request action can be represented by using an exclusive specialisation of two classes of objects (D and E), where depending on failure or success one of the objects will always be created. In the case of a failure, an object (D) is terminated by performer (Response 1) and search pattern can be again reiterated by requester. In case of a success, a requested object is found (E) and then reclassified by displaying it to a requester (Response 2). The object (C) is created.

The example with search pattern is represented in figure 63.

![Figure 63: Example of search pattern](image)
The search pattern in the presented example can be explained in the following way: the post-condition of Request Flight action is an exclusive specialization of two objects Flight Request[Not Fulfilled] and Flight Request[Fulfilled]. If the flight request was fulfilled, then the compositional object of Flight[Found] was created and Reservation flow is sent to Customer by invoking operation Create Reservation. If the Flight Request was not fulfilled then the flow with rejected request is sent to the Customer by invoking operation Reject Request. In that case, the Customer has possibility to reiterate the search again. Note: The object of Flight will not be terminated in the case where the request was rejected, because it will be necessary in the next interaction loops.

Various combinations of the static and dynamic dependencies are able to express the main workflow control patterns. Traditionally, only dynamic models are used in pattern analysis. The separation of the static and dynamic details of the presented patterns creates fundamental difficulties for two major reasons:

1. since the static details must somehow be compensated for by using dynamic constructs, the number of the basic patterns becomes bigger than is really necessary. Sometimes, their small differences are difficult to understand and visually unrecognisable by business experts.
2. if the static aspects are not taken into account, then patterns will become more complicated to use for the purpose of business process modelling and evolution, which is important for change management.

5.3. Transition Principles to Implementation-Specific Design

This section presents the research results concerning how to pass from computation-neutral conceptual modelling to implementation-specific design. It shows how service-oriented modelling can be mapped to object-oriented modelling. The service-oriented way of modelling could be seen as a way of motivating implementation-specific design. It enables system analysts and designers with the ability to verify implementation-specific design and validate their solutions to goals. Supporting technical system specifications can be justified in the context of service-oriented diagrams. The transition from computation-neutral modelling to implementation-specific design takes place between the semantic and syntactic modelling levels. This chapter presents transition principles, which are as follows:

- a coherent set of service interactions is mapped to one independent software component.
- service interaction flow structure must match one user interface class.
- service requester and service provider are mapped to one actor in UML use case diagram.
• every action in the service-oriented diagram corresponds to an object-oriented event, which should be related to one or more creation and termination effects. Every effect is defined by one object-oriented operation.

5.3.1. Extending Lyee Methodology Using Enterprise Modelling Approach

The International Joint Research Project was a three year project, financed by Catena Corporation in cooperation with Iwate prefectural University in Japan. Karlstad University was one of the partners of the project. The objectives of this project were to establish and expand a new software development method that would provide broader application of the Lyee Methodology.

The goal of the Karlstad University research group in this project was to extend the Lyee methodology by using pragmatic and semantic representations. The major contribution of the research done from a practical point of view was to demonstrate, by various examples, how semantic models can contribute in the phase of system requirements engineering in the context of Lyee. The overall investigation was based on examples of a real case study in the area of e-commerce (Gustas & Gustiené, 2002; Gustas et al., 2002).

Software system problems often occur because various applications do not fit or do not support the organisational components as was anticipated. Lyee methodology is one of the important attempts to overcome the software component fitness and integration problems, by designing software in a non-traditional way. The Lyee methodology (Negoro, 2001) is based on a way of building software on a basis of various logical structures of layouts and process route diagrams by which information system users are supported in performing their tasks (Gustas & Gustiené, 2002).

According to Lyee methodology, the first step in the requirements engineering process is to establish the set of data (objects or words) that an application should deal with (Poli et al., 2002). Words of the natural language sentences are used to make decisions on various screen, message, and printout and file layout structures. The logical layout structures are used as a basis for the definition of the Process Route Diagrams (PRD), (Negoro, 2001). Process Route Diagrams are used in the input of LyeeAll case tool, which is able to create a source code for a specific software application. The logical layouts and derivation rules (application logics) of specific concepts are coded using the conventional programming languages. The control logics of the software component are created by the LyeeAll according to a Process Route Diagram.

The most important contribution of the Lyee approach is that an internal application control is separated from the coding of the logical design and concept derivation rules. In conventional software development, integrity among all the three parts is responsibility of a designer and programmer; that is why software testing plays an important role in an overall development life cycle. By using Lyee approach, the internal application control is materialised through the scenario function. It is
derived automatically according to the process route diagram. This is illustrated by figure 64.

![Figure 64: Core structure of the Lyee methodology](image)

The weakest among all presented links in the core structure of the methodology is the way in which various rules and logical layout structures are captured. The logical design structures are formulated according to users’ requirements by using a natural language. Ambiguities, incompleteness and inconsistencies that may take place among logical layout structures are difficult to identify and resolve because an integrated view to the software system and organisational system architecture is not available. It may cause difficult decision problems on how various layout structures must be shaped. Typically, this is a very tough requirement for several reasons. Information system users are not able to present technical system requirements. They need to be guided to reach an understanding of how the logical design of the screen or file layouts should look like. Such an agreement cannot be reached without a complete understanding of how software application or the overall information system is conceptualized. According to the service-oriented modelling approach, one of the problems in most conventional software development approaches is that the implementation level requirements cannot be represented without taking into consideration the organisational requirements, because they are not able to address extremely important issues of design fitness.

One of the objectives of extending the Lyee methodology is the adoption of semantic models and using the service-oriented way of modelling for the conceptualization of IS architecture. The graphical representations are used for the purpose of visualisation and reasoning about the information system quality. Every enterprise component must be structurally and behaviourally motivated and consistent with respect to the whole. Analysis and graphical representations of enterprise architecture may help to resolve some IS inconsistency and integrity issues even before the layouts of screens, messages, printouts and files are designed as an input of LyeeAll case tool (Gustas et al., 2002). The core structure of the extended Lyee methodology framework is presented by figure 65.
Software systems should support business processes and software, and to be regarded as a value-added technology, they must fit various business processes. Change management in the organisational part or in the technical (software) part of the system is a big challenge, because even a simple deviation from the traditional business practice may be considered as a symptom for a new problem. The key issue is the determination of the true IT needs and how these needs are integrated into the overall organisational system. Enterprise architecture was designed using a set of service-oriented diagrams.

5.3.2. Bridging Computation-Neutral Modelling to Lyee Diagrams

To reach a coherent, complete and consistent graphical description of a new information system, the concept of enterprise should be analysed in a broader sense taking into account the description of pragmatic, semantic and syntactic aspects of the whole information system or some part of it. These three aspects are important for semantic modelling the ultimate goal of which is to introduce a common basis for the integration of enterprise architecture at the syntactic, semantic and pragmatic levels (Gustas et al., 2002). This idea is illustrated in figure 66.
In this project, the pragmatic level corresponds to strategic business descriptions. These descriptions determine various enterprise components at both the semantic level and the syntactic level. At the semantic level, static and dynamic aspects of the information system are defined, using semantic dependencies which are established to capture semantic details about various relationships among concepts. Enterprise models on three levels provide the ability to describe the semantics of the information system in a clear and understandable way. This is very important for information systems as well as software engineering processes.

The purpose of the research done in this project was to demonstrate how static and dynamic constructs for system requirements engineering can be applied in the context of Lyce methodology. The overall investigation is based on the example of a real case study that is taken from the area of e-commerce, which deals with buying trips on Internet.

A short example is presented to demonstrate how Lyce layouts are motivated. The examples show how the semantic part of the enterprise model is implemented and how the logical design can be used to define a Process Route Diagram (PRD) that is the main logical structure used in the input of the LyceAll tool. Enterprise models provide a basis for the gradual understanding of why and how various layouts, computation rules and process routes come about. The complete information about case study can be found in the annual project report (Gustas et al., 2002).

The demonstration begins with showing an electronic version of the service Trip Reservation process, which is a human-to-machine interaction process. This process needs to be supported by a software component. Figure 67 presents an initial description of the service Trip Reservation.

![Initial service description](image)

**Figure 67:** Initial service description
The presented description is not complete. It presents just three main steps (see section 5.2.5). At this stage, it presents the actors with value flows (step 1), identifies static dependencies between actors (step 2) and identifies actions (step 3). Any flow dependency may imply a communicative action. It is then considered to be an action and a communication flow. Cohesion of action and flow in semantic modelling results in a more complex abstraction, which is entitled as a communication action. The flow dependency link between two actors implies that these actors are dependent on each other not just by flow, but by the action as well.

The electronic way of the trip reservation process is constrained by other additional communication flows (in comparison with non-electronic way) such as selection of a trip from the Trip List and filling in Customer Identification Data. The diagram indicates that the User is entitled to send Trip Requirements to Internet Agent. The buttons on screen layouts (see fig.70) are mirroring the actions at the semantic level. When a button on the user interface is clicked (it will be implemented as a Find button on the screen layout in figure 70), the Find Trip action will be invoked. Escape button (see fig.70) indicates that the user can leave the application at any moment, invoking operation escape (see fig.68).

When Find Trip action is executed, Travel Agent invokes the action Show Trips and the Trip List, with available trips, are displayed to the User to select the trip. Selected Trip flow will be sent to Internet Agent, which will invoke the operation Show Registration Form. The diagram also presents the possibility of a contingent action. If the trip was not found according to user's requirements, the Travel Agent invokes operation Advise User, presenting the screen layout where the user has the possibility to continue searching for trips, or he can quit the session by clicking the Escape button. More about contingent actions and concerning screen lay outs can be found in the project report (Gustas et al., 2002).

If the action is invoked, it should mean some changes in the state of affairs; otherwise the action is not purposeful. The communication action can be viewed in two perspectives as state transition and as communication. Various concept states result from different communication actions. Any communication loop is able to change the static associations between instances. State changes are important to the actors. Without the ability to represent noteworthy state changes such as Trip [Found] or Trip [Selected] (see fig.68), it would be difficult to understand the rational and effect of every communication loop. Actions express the possible ways in which state changes may occur. These changes are specified by using the integrated dependency communication and transition. This is presented in figure 68.
Cohesion analysis is important for a semantic integrity control between a computerized information system state change and communication flow’s internal structure. Any parameter of a communication flow is supposed to be used for a certain purpose: it might be the creation of an object or the deletion of an object in some state. A removal communication action must destroy all associations that are relevant for this state. A creation action must have the capacity to establish all relevant associations, otherwise the action is incoherent. It should be noted that some actions can create and destroy associations at the same time. Such internal changes can be followed according to the static dependencies defined for various classes of objects. For example: Select Trip action creates the object Trip in state selected and terminates the object Trip in state found.

Communication flow parameters can be consumed or emitted by a predefined action. A consumption action is supposed to create values for the matching class attributes. Every consumed parameter has to be defined in a post-condition class by using the same or a different name. If this name is different, then an explicit computation rule must be defined. Usage of any attribute should be questionable with respect to an overall structure of incoming flow. If an attribute is irrelevant for the action, then it should be removed. The termination process can be represented by an action without a resulting state. In this case, no commitments of actors should be pending in connection to this state. A properly designed termination action should
emit values of class attributes by a communication flow that can be viewed as a message to the service requester.

The communication flow structure and the class attributes define requirements for the specification of syntactic elements of the Lyee methodology. The LyeeAll2-tool requires a definition of all identifiers that are used in the development of the application. For instance, the dependent concepts of Trip constitute the foundation of a file layout that can be represented by the following relation:

\[ \text{TRIP} \ (\text{Trip Number}, \text{Destination}, \text{Airport}, \text{Leaving Date}, \text{Leaving Time}, \text{Returning Date}, \text{Returning Time}, \text{Number of Trips}, \text{Price}) \]

When Find Trip action receives parameters from a User, a Trip List object is created, which consists of one or many Found Trips, otherwise, the Find Trip action fails. Any Found Trip should satisfy the specific computational rules. The action is supposed to match the received parameter values with Trip List class attributes. The Trip Requirements communication flow structure is presented by figure 69.

![Trip Requirements flow structure](image)

**Figure 69:** Trip Requirements flow structure

This structure, with all possible action names that could affect the illustrated communication flow parameters, defines the static requirements of the screen layout. The example of the screen layout Trip Requirements is represented in figure 70.
Various constraints that need to be implemented by the software component (Jakobsson, 2002) in the Lyee approach can be represented as Logical Formulas. At the computation-neutral modelling level, those constraints are defined as the computation rules. A few examples of the computation rules, which were identified for a Find Trip action, are as follows:

- Trip [Found].LeavingDate = Trip Requirements.Date
- Trip [Found].ReturningDate = Trip Requirements.Date + Trip Requirements.Length of Stay
- Trip [Found].Destination = Trip Requirements.Place.To
- Trip [Found].Airport = Trip Requirements.Place.From

Here: ‘.’ is the prefixing operation. It is used for the unambiguous identification of attributes.

An example of an inference rule applied is the following:

If A → B, and B → C, then A → C.

If Trip [Found] → Trip, Trip → Leaving Date, then Trip [Found] → Leaving Date

This inference rule is very important as it helps to identify and eliminate undesirable characteristic of redundancy.

The syntactic level of the Lyee methodology is based on the printout, message, screen and file layouts. These primitives and their dependencies are elements that are used in the definition of a so-called Process Route Diagram. It can be used as a definition of a logical structure for a software source code that is built by the LyeeAll tool. Syntactic elements are considered to be the basic building blocks that define an implementation-specific perspective. These elements can be defined by using the conventional programming languages or database definition languages. For example,
database relations are typical representations of the syntactic level. A technical component can be a machine or a software component. A coherent set of interactions is delegated to one independent component. Service interactions can be defined in terms of components and their interfaces with the structural definition of the printout, message, screen or file layouts. Notations of interfaces between components in the Lyee approach are represented in figure 71.

![Figure 71: Notation of interfaces for Lyee approach](image)

Screen forms and printout forms are used for interaction flows between organisational and software components. Message layouts are designed for interaction between software and hardware components. Humans, technical systems or organisations can be thought of as enterprise actors that are dependent on various information, decision or material flows. Flow dependencies among actors are regarded as basic communication patterns, which, together with the actor, composition, or generalisation links, describe software system architecture (Jakobsson, 2009). Actors can be viewed as organisational or technical system components. An instance of an organisational component can be a human, group of people, a job-role, a position, an organisation, etc. Typical instances of the actors that are relevant for system development in the Lyee environment are represented by figure 72.

![Figure 72: Basic components for definition of enterprise architecture](image)

In this short presentation, it was demonstrated how the TRIP file layout and the screen layout for Trip Reservations was defined by semantic descriptions of service architecture. These layouts define the operating infrastructure of the Internet Agent software component. A relevant part of the Internet Agent enterprise model at the syntactic level is represented in figure 73.
This diagram demonstrates how the semantic part of the enterprise model, which was presented in figure 68, is supposed to represent implementation-specific design. It should be noted that various previously defined semantic dependencies have to be taken into account at the syntactic level. This means that they have to be consistent on various levels of abstraction. A special inference rule can be used (Gustas, 2010) in order to validate the semantic consistency of graphical representations. In this case, the service-oriented modelling framework can be used as a uniform basis of reasoning about the unambiguity, coherence and completeness of software processes that are defined on the neighbouring levels of abstraction.

The presented implementation-oriented diagrams, with syntactic elements for the definition of implementation perspectives, were used for the derivation of Process Route Diagrams (PRD), which are the basic syntactic level representations of the Lyee methodology. A PRD specifies the navigational links among various building blocks for generations of software components by LyeeAll tool. It should be noted that understanding and designing a PRD from the natural language description is a
very complicated task. The service-oriented modelling method can help system
designers to understand how and why a specific configuration of PRD is designed.
A PRD represents a basic behaviour of the software component. This behaviour is
rooted in a structurally richer specification than is shown in the previous diagram. A
corresponding PRD is represented in figure 74.

The represented part of PRD shows the files and main interactions in terms of
screens that are designed for the graphical interface. The diagram contains five
scenario function triples. Each of them consists of three pallets. WO4 is dealing with
the display of information flow to a user or storing information to a file or a database.
WO2 is responsible for accessing information and executing the computation rules.
The third pallet, WO3, deals with the recovery of input information flows and
navigating to the successive triples of the scenario function.

The pallets are composed of words and grouped into logical units on the basis of
the enterprise model at the semantic level. Specification of the logical units serves as a
software component logical definition in the LyceAll tool. Logical units define the
behaviour of the screen, file, message or printout layouts. The logical units are
identified according to the following rules:

---

Figure 74: A part of the Process Route diagram

---

132
SR – reading data parameter structure from a screen layout,
SW – writing data parameter structure to a screen layout,
FR – file retrieval parameter structure,
FW – data modification parameter structure,
KR - database query parameter structure (select statement),
KW – database query modification parameters (update or delete statement).

From a software component builder’s perspective, the definition of the logical units is not sufficient. The builder needs to understand the technical system and organisational architecture, and where the application is going to be installed. The service-oriented modelling method helps to understand not only pragmatic and semantic aspects, but also the design of software components.

Mapping between enterprise modelling and PRD is the only way to deal with the problems of integrity and consistency of a software system being developed with Lyee methodology. With the suggested extension to the PRD, it is possible to map between enterprise modelling and PRD without losing semantic details represented in Enterprise Architecture.

The goal with this example was to show how the syntactic elements of Lyee methodology can be engineered using the enterprise modelling approach (Gustas et al., 2002). It is a new systematic way of bridging conceptual implementation-independent modelling and computation-specific modelling. This way of modelling facilitates the justification and motivation of the functionality of software components that are used to support the work of organisational actors involved in various business processes (Gustas & Gustiené, 2002).

5.3.3. From Conceptual Representations of Services to Object-Oriented Diagrams

The objective of this section is to present the service-oriented modelling method, with an emphasis on showing how service-oriented constructs can be explained and mapped to component modelling and object-oriented diagrams (Gustiené & Gustas, 2008). The comparison of the basic events used in object-oriented analysis and design and in service-oriented analysis and design is also presented in this section (Gustas & Gustiené, 2009a).

The service-oriented way of modelling can be seen as a technique for justification of system design. It can help to check consistency and completeness of design specifications. Service-oriented diagrams can be incrementally refined into component-based representations, which are projected into a set of object-oriented diagrams. This technique could be useful both for business experts and system designers, as it provides a method for verification of consistency and completeness between specifications on three levels of abstraction. The graphical representation of these three levels is presented in figure 75.
Service-oriented architecture (Erl, 2005) represents a set of design principles (Kraffzig et al., 2005) enables business processes to be analysed in terms of services. The concept of service can be applied both to organisational and technical components, which can be viewed as service requesters and service providers (Gustiené & Gustas, 2008). The conceptual representation of service architecture takes place on a computation-neutral modelling level. The internal and external behaviour of a service can be explicitly defined using the principle of an interaction loop or a composition of loops between actors. Services can be represented in terms of events. Events are important as they serve to mark points in time when state changes occur. Every occurrence of an event is a noteworthy change in state (Martin & Odell, 1998). State changes are important because they represent changes of the objects in a problem domain and they signal a need to acknowledge the change in some way. Objects, at any moment, exist in a certain state. A state exists in one event, which caused the transition in some state, until another event occurs that causes a transition out of the state (Brown, 2002).

According to the service-oriented method, state changes define the objective perspective, which is complimentary to the intersubjective perspective (Gustas & Gustiené, 2009b). The objective perspective that defines the object's lifecycle in service-oriented diagrams is represented by using initial, intermediate and final classes, which are analysed in the context of interaction loops among actors. From the objective perspective, a request and response actions change business data from one consistent state to another (Gustas & Gustiené, 2009a). Moving flows together with request and response actions, which create or terminate objects in various classes, are critical to understand the semantics of services.

Figure 75: Three levels of the method
There are two basic events in the service-oriented modelling method: creation and termination (Gustas & Gustiené, 2009b), compared to six types of basic events used for object-oriented analysis and design (Martin & Odell, 1998). These two events are fundamental for the definition of the third important event; that is the reclassification event. A reclassification event can be understood as a communication action (Dietz, 2001). A communication action is able to express the semantics of all six types of object-oriented events; additionally, service-oriented modelling construct can specify pre-condition and post-condition object class associations, which are sufficient to visualize the event of disconnection and connection. An action is defined as a transition (→) from the pre-condition object class to the post-condition object class. Fundamentally, two kinds of changes occur during any transition: the removal of an object from a pre-condition class and the creation of an object in a post-condition class. Sometimes, objects pass several classes and are then destroyed. It should be noted that either the pre-condition or the post-condition class may be missing. The internal changes of services are defined using transition links. For instance, a request action is supposed to remove an object from a pre-condition class and to create an object in an intermediate class. The intermediate class is necessary to initiation of response action by the performer, which is then supposed to remove the intermediate data values and to create a post-condition class data.

The effect of any response or request is a reclassification, removal or creation of an object. Service requests and service provisions within a value chain or within a business process are defined in terms of basic events; creation, termination and reclassification. These three types of actions are used for the conceptualisation of a continuous or finite lifecycle of objects. The noteworthy semantic difference can be represented by a set of attributes or states that are associated to the pre-condition and post-condition class (Gustas & Gustiené, 2009a; Gustiené & Gustas, 2008).

The conceptual representation of services can be defined using the basic service-oriented constructs. The composition of different interaction loops results in more complex representations, which can be expressed by various service-oriented analysis patterns (Gustas & Gustiené, 2009b). Figure 76 presents the conceptualization of Recruitment Management Service process, showing the interplay of three basic constructs, expressed by a single service-oriented diagram.
The interplay between the three constructs provides an integration of the static and dynamic aspects. Such an integrated diagram not only represents actions, but also defines static dependencies on classes of objects, which must hold for triggering of these actions (Gustas & Gustienė, 2009a). The diagram presented in figure 76 demonstrates how sequences of communication actions prescribe creation, termination or reclassification operations (a reclassification operation includes both termination and creation operations). A person applies for employment by sending application (flow Application Data) to CEO of organisation. If CEO receives the application, then an object of Application and an associated object of Applicant are created (see composition link). According to the semantics of basic constructs, CEO is obliged to either employ an applicant or to reject an application. Note that both actions predefine the removal of an application object. If CEO decides to reject application, then an applicant is terminated. If he is employed, an Applicant object is reclassified to Employee by Employ action, which is exclusive to Reject action. Note that an Employee is a specialisation of a Person concept. Employee concept is characterised by the additional attributes of Position and Employment. Since Employee is a Person, the attributes Name and SS Number must be instantiated at the time or before an Applicant is created. These attributes are essential to characterise the semantic difference between Applicant and Employee. If an Employee would be terminated by some action, then the association links to Position and to Employment objects must be removed.

The represented semantic description of Recruitment service can be implemented by using a set of loosely-coupled organisational and technical components. An organisational system is usually supported by a technical system part, which can be conceptualized in terms of any software or hardware component. Typically, a coherent set of interactions are mapped to one independent technical component.
coherent interactions should fit together to achieve a common goal. Interactions of one technical and two organisational components are represented in figure 77.

![Diagram](image)

**Figure 77**: Recruitment management service as software component

Figure 77 presents the core business process of Recruitment Management Service, not showing the alternative *Reject* action. Representation of service in figure 77 is consistent with the conceptual representation of service presented in figure 76. In this example, a coherent set of interactions is supported by one software component, which is called Recruitment Management Service (Gustiené & Gustas, 2008).

Both object-oriented approach and UML are widely used as standards in many companies. Therefore, it is important to show how UML diagrams can be justified using a computation-neutral way of modelling. This section presents bridging rules from service-oriented conceptual models, used at a computation-neutral level, to UML diagrams, such as use case, activity and class diagrams. Figure 78 presents UML use case diagram for the same Recruitment Management Service.
Any communication action can be considered as a separate use case in the use case diagram. According to our example, if the `Apply` action is triggered, then two different outcomes are possible: either `Employ`, or `Reject`. According to the service-oriented diagram, one of the successive actions must always take place. Such details are not included in the presented use case diagram and therefore, for the purpose of consistency, another diagram should be used to express the missing semantics.

Use cases represent the functionality that a software component provides by interacting with actors. Organisational agents and recipients involved in any communication loop do not belong to the technical system part; therefore they are viewed as external actors in use case diagram. The specification of a use case diagram according to conceptual representation of a service is as follows:

- communication action is represented as a use case,
- software component, which plays the role of service provider, defines the service boundary as a technical service. In the use case diagram it is represented with a system boundary,
- service requester is represented as an external actor in the use case diagram.

Use cases can be decomposed into the component layer actions by using the `<<include>>` and `<<extend>>` relationship. This is represented in figure 79. `Apply` use case is decomposed into two use cases `Send Application Data` and `Receive Application Data`. `Employ` use case is decomposed into `Send Employment Data` and `Receive Employment Data`. 

![Use case diagram](image-url)
The semantics of a use case can be represented by using sequence and activity diagrams. We will limit the process view examples to activity diagrams. The activity diagram of Recruitment Management Service is presented in figure 80.

The activity diagram presents process logics as well as showing the mutual exclusives of use cases, the details that were missing in the use case diagram. The object-oriented operations, which define a use case, can be elicited from the service-oriented diagrams. A method for implementation of the Apply use case is defined by using UML activity diagram, which is presented in figure 81.
The method of the Apply use case must include two interface operations: Send Application and Receive Application. The Send Application operation should trigger both Create Applicant and Create Application operations. According to the semantics of service-oriented events, the Receive Application operation is executed together with a Change Application Status operation that is initialising state of an Application object with the status Received. Use case Employ consists of two interface operations: Employ Applicant and Receive Employment. The remaining domain operations are predefined by the service description as well. The corresponding UML activity diagram is represented in figure 82.

**Figure 81:** Method of the Apply use case
Internal changes of services (Hull et al., 2003) are represented by the state transition links. Transitions are triggered by operations, which specify the permissible ways for changes to occur in different classes of objects. State changes are typically specified using a finite state machine, which describes the sequence of operations that occur in response to events (Gustas & Gustienė, 2009a). A graphical example of a state transition diagram is represented in figure 83.

The state of an object can be defined as a collection of associations an object has with other types of objects and their attribute values. There are six types of basic events (Martin & Odell, 1998) that are relevant for object-oriented analysis and design. These events are as following:

- **creation** typically triggers an operation, which creates a new object in a class
• **termination** is the opposite of creation. It triggers an object removal operation,
• **a connection** event adds an entirely new association between objects. The connection event can also be used to describe changes of an attribute values,
• **a disconnection** event removes an existing association between two objects,
• **classification** triggers the creation of an object in a more specialized class,
• **a disconnection** event removes an existing association between two objects.

**Creation** and **termination** events can be represented by transitions from an initial state to final state. The termination event can be expressed using removal operation, which consumes an input object into a final state. A creation event is defined using creation operation, which produces an output object in an initial state. A graphical representation of termination and creation events is presented in figure 84.

![Termination and creation events](image)

**Figure 84:** Termination and creation events

**Connection** and **disconnection** events may correspond to an update operation that is associated with the state transition. These events can be implemented using a sequence of object creation and termination operations. To perform a state change, an object needs to be accessed. A state change can be represented by an update of one or more attribute values. The semantics of a state change can be defined by a reconnection operation, which is realised as an update of one or more attribute values. A reconnection event pattern is presented in figure 85.

![Reconnection event](image)

**Figure 85:** Reconnection event

The problem is that a reconnection event pattern can not be uniquely defined by using object-oriented diagrams. It cannot be visually recognized by a system designer simply from the diagram. The same pattern is used to represent other object-oriented event types such as connection and disconnection. A sequence of two events requires an intermediate state, which makes no sense for someone who has little or no knowledge in the area of object-oriented design. Such artificial states are implementation-oriented details, which add complexity. An intermediate state is redundant and it cannot be justified by a business analysis expert. Regrettably, object-oriented approach and UML do not support the modelling of dynamic classification (Maciaszek, 2005). This one of the weaknesses of the object-oriented approach, as in reality, objects do change classes dynamically.
The termination and creation operations are used for the implementation of *declassification and classification*. Reclassification is a compound event that should be viewed as a simultaneous declassification and classification. The semantics of reclassification are quite comprehensible for business experts. Nevertheless, a reclassification event has no easily recognisable counterpart in object-oriented models. The method of reclassification, declassification and classification from Class1 to Class2 can be shortly characterised by the following steps (Gustas & Gustiené, 2009a):

- the creation of an object in Class2 by copying all attribute values from Class1 to a newly created object in Class2,
- the disconnection of all associations pointing to the old object and connection of them to the new one in Class2,
- the removal of the old object in Class1.

The described sequence of operations artificially adds complexity by multiplying a number of states, which cannot be justified from a system analysis point of view. Increasing the complexity of state diagrams increases the complexity of IS analysis phase in general.

The problem is that object-oriented analysis and design constructs, which are implementation-dependent, are used in the system analysis and design phases. When using such constructs at an analysis phase, where analysis techniques are intended to be independent of technical solutions (Fowler, 1997), such constructs cause difficulties in the validation of graphical representations of the IS by business process analysts, thereby making IS specifications prone to inconsistencies, discontinuities and ambiguities. It results in the failure of business experts to recognise the noteworthy events, which results in state changes. The system analysis process should be based on a computation-neutral modelling notation. It would make communication among system analysts and system designers more comprehensible and productive.

Static aspects are crucial for understanding the semantics of the basic events. Nevertheless, the consequences of state changes are difficult to follow in the presented object-oriented patterns. The problem is that the static and dynamic aspects are disparate, because they are represented in different diagram types.

Unfortunately, there is no guide to how semantic integrity control of the static and dynamic aspects can be performed in UML models. If the same semantic patterns are used for the description of different basic events, then a visual analysis of diagrams is very difficult. There is no way for system analysts to understand the consequences of changes other than to manually explore a method of every operation, as automatic recognition of connection, disconnection and reconnection events is impossible.

The pre-condition and post-condition object classes are defined by the service description and can be implemented in a number of ways. These classes prescribe dependencies for a class diagram design. In the presented example, all service description classes are viewed as independent UML domain classes (Gustiené & Gustas, 2008). The class diagram is presented in figure 86.
The conceptual representation of Recruitment Management Service prescribes two types of interface classes – one for a Person and one for CEO. For instance, Send Application and Receive Employment operations must be included into Interface Person class. Receive Application and Employ Applicant operations are defined in the Interface CEO class.

If CEO decides to employ an applicant, then Employ Applicant operation is triggered in the Interface CEO class. According to the presented service description, Employ Applicant action requires both Create Employee and Delete Applicant. The creation of a new Employee object requires the creation of an Employment class object as well. That is why the Create Employee operation is defined in a sequence with the Create Employment operation. Since an Applicant is composed of an Application, the creation of an Applicant object is synchronised with creation of an Application object as well (see Delete Application, Delete Applicant and Create Applicant and Create Application operations in both activity diagrams). The communication loop is completed when a person receives Employment Data. This information flow is provided by the Receive Employment operation, which is placed in the Interface Person class. As it is prescribed by the service-oriented diagram, Receive Employment is executed in sequence with Change Employment Status operation. In general, if the termination event takes place, then all objects in more specific classes are terminated as well (see inheritance links). This rule is not relevant for the objects of more generic classes. If an object is terminated in a more specific class, then objects of the more generic classes are still preserved.
An integrated method was used to show a possible way to map from the service-oriented conceptual representations to the component level and to object-oriented diagrams. The theoretical method was illustrated with an example to show how service-oriented constructs predefine the semantic details that were used for the elicitation of the object-oriented operations. Such a hierarchical way of modelling provides with a consecutive order of actions to justify and understand the application logics. The research presented in this chapter contributes with a method of how to pass systematically from computation-neutral modelling level to implementation-specific design. Conceptual representations of service architectures done at a semantic level using the service-oriented way of modelling can be used for consistency verification of object-oriented diagrams. It could be also used to validate service architectures, as goals stated at the pragmatic level and analysed at the semantic level should be consistent with design specifications. In comparison with object-oriented basic events, service-oriented events that form an integrated service-oriented construct to represent the semantics of object state changes are more comprehensible and can be communicated among stakeholders more effectively than a set of various types of object-oriented diagrams.
Chapter 6

Contribution and Conclusions

6.1. Publications and their Methodological Overview

In this chapter, the overview of all publications and the summary of the research results presented in the publications are presented. Publications 4, 5, 6 and 11 are book chapters. Publications 1, 2, 3, 7, 8, 9, 10 are peer-reviewed articles in conference proceedings. The overview of all publications, including the most important methodological aspects for the research study (research question, research method, premises for the research, main concepts and observations, contribution, implications and responsibility for the papers), are presented in the tables below. This chapter also presents a short summary of all the research results. Full papers are presented in Part II. The research results presented in each paper contributed to the overall new knowledge presented in the thesis. They contributed to the development of a new service-oriented modelling method for information systems analysis and design.


Table 6: Distinguished characteristics of publication number 1

<table>
<thead>
<tr>
<th>Authors</th>
<th>P. Gustiené, R. Gustas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How to cope with semantic problems of ambiguity in the area of conceptual modelling? The paper questions whether it is reasonable to set up a strict classification of semantic roles and how to cope with the problem of role relativity by introducing a set of semantic dependencies</td>
</tr>
<tr>
<td>Research method</td>
<td>The study of theories by various authors, who have contributed to the area of semantic role analysis and comparison of semantic roles used in different models. Conceptual meta-models of two different semantic role classifications were built for the purpose of comparison that helped us to detect the causes of ambiguity in conventional modelling methods. A simple example was used to demonstrate the ambiguity problem detection and relativity of semantic roles in conceptual modelling.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>The strict classification of semantic roles can be identified in most conventional approaches of information systems. The strict classification of concepts is a cause of structurally different, but semantically equivalent, representations. The relativity of semantic roles is an intrinsic property of the most conventional approaches of information system analysis and design. Information system developers use various graphical languages as well as natural language to represent and explain their solutions. It is not unusual that a natural language text may appear ambiguous, incomplete and inconsistent with respect to the diagrammatic solutions it is supposed to clarify. Relevance</td>
</tr>
</tbody>
</table>
of information is a relative issue. The same message, depending on a situation, can be relevant or irrelevant for the same user. The condition, under which an information system developer knows what information is needed for the user, is when he has a complete understanding of an overall business activity. None of the semantic modelling approaches have clearly defined guidelines on how to cope with the semantic ambiguity of object classes, which are processed by various communication actions.

Semantic modelling approaches do not take into consideration a notion of the context in which communication interaction takes place. The inability to represent contextual modifications in conceptual modelling is one of the major factors in misunderstanding that causes information systems development failures.

| Main concepts and observations | Main concepts: ambiguity, concept modelling, semantic roles, relativity. Observations: Semantic ambiguity is at the centre of the studies on human cognition and conceptualisation. The meaning is a multifaceted notion: the messages can be ambiguous because they may be understood in different ways. Ambiguity of the meaning may occur because the sentence is inconsistent or merely vague. One way to find out the causes of ambiguity is by studying the conceptual models in different contexts, which that are expressed in terms of communication flows between the different actors involved. Most of the conventional information system development methodologies neglect the essential semantic aspects of communication. Therefore, their semantic modelling techniques have no well-defined criteria that tell system designers whether the conceptual structure of communication flow between actors is sufficient with respect to the domain of reference. In contrast to the traditional semantic models, the agent-based approaches put into the foreground a communication aspect. This kind of knowledge is crucial to explain about unambiguity, consistency and completeness of conceptual models. The inability to link interactions and object transformations in conceptual modelling causes various problems in the areas of view integration and information systems analysis. The relativity of semantic role classification is just one cause of the semantic conflicts of diagrams that result in information systems development project failures. |
| Contribution | My contribution to this research was the analysis of these semantic roles that helped to identify the problem of semantic role relativity, which led to the conclusion that it is impossible to set up a strict classification of semantic roles, because it hides important semantic details of conceptualizations. A strict classification of semantic roles may hide important semantic details of enterprise architectures. It causes structurally different, but semantically equivalent, representations. It was found that changes in the interpretation of semantic roles are constrained by the nature of the pragmatic and semantic dependencies through which the concepts are connected. It was demonstrated that a set of static and dynamic dependencies can be viewed as a modelling technique to define and analyse a deep semantic relationship among concepts. Conceptual models of message flows can be used to clarify the reason of ambiguity. It has been shown that enterprise modelling approach, which is used in the area of information system analysis, can handle this problem in a more flexible way. It was demonstrated how conceptual models of messages helped to clarify the reasons of ambiguity. |
| Implications | Analysis of semantic role relativity is also applicable for most conventional approaches of information system analysis. Information system methodologies should support flexible interpretation of semantic roles, because strict |
classification of concepts can be a cause of structurally different, but semantically equivalent, representations. In enterprise modelling approach, the interpretation of semantic roles is flexible. Whether a concept is regarded as an instance, class, communication flow, state, problem, goal or an actor depends upon the type of semantic links that relate it to other concepts. Semantic dependencies can be viewed as a modelling technique to define and analyse deep semantic relationships among concepts.

The enterprise modelling framework can be used as a uniform basis of reasoning about the ambiguity and integrity of diagrams that are defined on the various levels of abstraction. Enterprise models are intended to support reasoning and to validate information system requirements before they are implemented.

| Responsibility | Prima Gustiené 70% | Remigijus Gustas 30% |


| Authors | R. Gustas, P. Gustiené |
| Research question | How to extend the Lyee software engineering methodology by introducing early requirements specifications?  
| | How to adopt a systematic analysis of pragmatic and semantic descriptions for the analysis of logical software specifications in the context of Lyee? |
| Research method | The enterprise modelling approach was used in the case study to demonstrate how pragmatic, semantic and logical Lyee software requirements can be defined and integrated.  
| | Graphical representations were applied for a system analysis of both traditional and electronic business processes on a Trip Agency case study. |
| Premises for the research | Software system problems very often occur because various applications are not able to support the organisational components as was anticipated. There is no other way to study the organisational and technical process fitness when the new software components are introduced, without describing enterprise architecture of an existing system prior to designing a new one. Explicit representation of the organisational and technical system architecture is a necessary condition to understand orderly transformations of the existing work practices. Most of the software methodologies are heavily centred on system development issues at the implementation level. Such methodologies are restrictive in a way that a supporting technical system specification cannot be justified in the context of organisational process models. The most difficult part of enterprise modelling is to reach a coherent, complete and consistent graphical description of a new information system.  
| | The presented modelling method is aimed at defining enterprise system in a broad sense, taking into account pragmatic, semantic and syntactic aspects of the whole information system or a part of it. The Lyee approach is built on the assumption that software developers are provided with logical structures of layouts by which information system users are supported in performing their tasks. In many cases, information system users are not able to present the technical system requirements in this way for several reasons. They need a systematic guide for achieving a mutual agreement on how the logical design of the screen or file layouts should look. Such an agreement cannot be
reached without a complete understanding of how the overall information system is working. A long list of software system development failures demonstrates that the assumption that information system users are IT experts is not correct. Most customers are not able to describe precisely the requirements of how a design of the screen and file logical layout structure should look.

The main effort of this research study is to demonstrate a constructive way for the extension of Lyee software engineering methodology using enterprise modelling approach and to provide guidance on how to control the consistency of system specifications that are presented on various levels of abstractions. The integration of pragmatic, semantic and logical software requirements using enterprise models is the main issue in this paper.

<table>
<thead>
<tr>
<th>Main concepts and observations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main concepts</strong>: Enterprise modelling, semantic level, pragmatic level, syntactic level, semantic and pragmatic dependencies.</td>
</tr>
<tr>
<td><strong>Observations</strong>: A starting point of the Lyee requirement engineering process is system specification in natural language, which is used to make decisions on various screen, message, printouts and file layout structures. The most important contribution of Lyee methodology is that an internal application control is separated from the coding of the logical design and concept derivation rules. In the conventional software development, integrity among all these parts is the responsibility of a designer and programmer. That is why software testing plays an important role in an overall development life cycle. Using the Lyee approach, the internal application control is materialized through the scenario function and it is derived automatically according to the process route diagram. The weakest point in this methodology is how various rules and logical layout structures are captured. The logical design structures are formulated according to users’ requirements using natural language. Anomalies, incompleteness and inconsistency that may take place among logical layout structures are difficult to identify and resolve because an integrated view to software system and organisational system architecture is not available. It may cause decision problems on how various layout structures must be shaped. The key issue is the determination of the true IT needs and how these needs are integrated into the overall organisational system. In many cases, information system users are not able to present technical system requirements. Logical design cannot be reached without a complete understanding of how the overall information system is working. The ultimate goal of the enterprise modelling is to introduce a common basis for the integration of the static and dynamic aspects of communication. It aims for a description of pragmatic, semantic and syntactic aspects of the whole information system or some part of it. Enterprise modelling language can serve as a starting point for graphical representations of users’ requirements all the way across organisational and software system boundaries.</td>
</tr>
</tbody>
</table>
The integration of the pragmatic, semantic and logical software requirements using the enterprise models is the main contribution of the presented research. It was demonstrated that enterprise models provide a basis for the gradual understanding of why and how various layouts, computation rules and process route diagrams come about. It was also demonstrated how the syntactic elements of Lyee methodology can be justified using the enterprise modelling approach. It was demonstrated that enterprise modelling framework (that consists of three levels) can be used as a uniform basis of reasoning about ambiguity, coherence and completeness of software processes that are defined on the neighbouring levels of abstraction. This way of modelling suggests a new way of defining the enterprise architecture, which is based on an intentional way of thinking and on a new way of reasoning. My research contributed to the extension of Lyee framework with a way of modelling of service interactions. The case study contributed with the modelling of pragmatic aspects and transition principles from computation-neutral to implementation-specific design.

Enterprise modelling language can be used as a starting point for the graphical representation of user requirements. Enterprise models can be used for change analysis in a systematic way on the basis of the graphical representations that are defined for both manual and computerized business processes. They provide a new way of defining enterprise architecture, which is based on a new way of reasoning. Graphical representations can be used for the purpose of visualization and reasoning about information system quality. Every enterprise component must be structurally and behaviourally explained. Enterprise models can be considered as a corporate resource in diagnosing potential problems. These models are crucial to enable reasoning about business process integrity and the purposeful implications of organisational change. The service-oriented modelling approach can be used as a core method to analyse the rationale of the new organisational solutions. It should help business managers, system developers and IT experts to define visualize and assess various organisational changes by using a graphical approach to business process modelling.

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>R. Gustas 60%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P. Gustiené 40%</td>
</tr>
</tbody>
</table>


Table 8: Distinguished characteristics of publication number 3

<table>
<thead>
<tr>
<th>Author</th>
<th>P. Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>Improving the semantic quality of system specifications by applying semantic dependencies and inference rules. How can undesirable characteristics be identified in different types of diagrams on different levels of abstraction?</td>
</tr>
<tr>
<td>Research method</td>
<td>Semantic dependencies and inference rules were applied for the identification of such undesirable characteristics of system specifications as redundancy and ambiguity.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>One of the problems in enterprise engineering is the lack of criteria for inconsistency detection. Graphical representations that are used for the visualization of different parts of IS specifications should be characterised by</td>
</tr>
</tbody>
</table>
such system qualities as consistency, coherence, un-redundancy, completeness and unambiguity. Ambiguity detection is especially relevant at the pragmatic level, when the representations of system requirements are fuzzy. It is necessary to have certain criteria for the identification of undesirable characteristics in different graphical representations on different levels of abstraction. Inference rules also are necessary for the identification and elimination of undesirable system qualities.

<table>
<thead>
<tr>
<th>Main concepts and observations</th>
<th>Main concepts: Quality control, ambiguity, redundancy, inference rules, semantic dependencies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
<td>Enterprise engineering is the process dealing with modelling and integration of various organisational and technical aspects of conceptualisation. It is a prerequisite for enterprise integration and reengineering. One of the problems of enterprise engineering that there is no criteria for how to check the inconsistency of system conceptualisations. The quality of system specifications is defined as internal consistency and correctness. A prerequisite for it is that the semantics and syntax of the concepts used by a particular methodology should be unambiguously defined. Certain rules are necessary to explain how to identify and eliminate undesirable characteristics in order to make system specifications more complete and consistent. Semantic consistency between the specifications of semantic and syntactic levels, as well as completeness of conceptualizations, with respect to pragmatic specifications, is critical in enterprise engineering. Enterprise modelling using static semantic dependencies can help detect semantic inconsistencies of IS specifications.</td>
</tr>
</tbody>
</table>

| Contribution                  | Semantic normalisation of the concept model was performed due to the inference rules to discover and eliminate a set of redundant dependencies. Two cases of redundancy were traced using the inference rule: redundant inheritance dependency and redundant set of attributes used in the model. One case of ambiguity was traced analysing the conceptual structure of the concepts. Modelling mechanisms such as inheritance and classification were used to analyse the structure of the concepts and reduce semantic ambiguity. The refinement of ambiguous concepts during the enterprise modelling process might help to sort out problems of misunderstanding. It was demonstrated that the quality of system specifications can be increased by using semantic dependencies and inference rules. |

| Implications                  | To achieve a high quality of IS specifications, it is necessary to have a way to identify and overcome undesirable characteristics of system specifications. System analysis means the creation of the shared set of concepts, which form a basis for successful communication among stakeholders. Semantically clean IS specifications should be defined in terms of unambiguous concepts. An analysis of semantic dependencies between concepts and usage of inference rules showed the way to identify undesirable qualities of system specifications. |

| Responsibility                | P. Guttienė 100% |
**Table 9:** Distinguished characteristics of publication number 4

<table>
<thead>
<tr>
<th>Author</th>
<th>Prima Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How to bridge organisational and technical system conceptualisations by using enterprise modelling approach? How to overcome inconsistency problems in system specifications?</td>
</tr>
<tr>
<td>Research method</td>
<td>The investigation is based on the examples of a real case study from the area of e-commerce. Case study was done using enterprise modelling approach to demonstrate the usage of semantic specifications for information system analysis and design.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Development of information systems involves not only organisational changes such as changes in work practices, but also changes in the technical parts of the system. Adapting to changes in the market means changing business models. Every change requires a re-engineering of communication patterns among human and technical components. It is necessary to have the possibility to identify the conflicts between organisational and technical system parts in a systematic way. Very often these systems are analysed separately. Model-driven development can be seen as possibility to provide solutions for solving such problems.</td>
</tr>
<tr>
<td>Main concepts and observations</td>
<td><strong>Main concepts:</strong> Enterprise modelling, enterprise modelling approach, model-driven approach. <strong>Observations:</strong> Model-driven development follows the idea that the complexity of information systems is driven by the complexity of the underlying real world. Paying more attention to modelling of an enterprise as a problem domain prior to implementation issues, information systems developers can better manage the complexity of the developed system. It is necessary to have a method that could help system architects to control the refinement process of system specifications on different levels of abstraction. The model-driven approach provides a separation of concerns for different levels that are important for the separation of business requirements stated at the pragmatic level and functionality requirements stated at the semantic level.</td>
</tr>
<tr>
<td>Contribution</td>
<td>The main contribution of this study was showing how the enterprise models can be used for change analysis in a systematic way based on the graphical representations that are defined for both traditional and electronic business. It determines the usage of enterprise models that should be used as a core method for the analysis of organisational solutions. It was shown how the syntactic elements of implementation can be justified. Enterprise models were used as a method for analysis of the organisational solutions before implementation solutions were introduced. It demonstrated that enterprise models are relevant to reason about business process integrity.</td>
</tr>
</tbody>
</table>
Implications

Enterprise models can be used as a means for change analysis in a systematic way on a basis of graphical representations. Such graphical representations should help business people as well as system designers to determine the fitness of various organisational changers. Using a systematic enterprise modelling approach, it is possible to bridge the organisational and the technical parts of the system. Enterprise models can be seen as a means for reasoning about consistency of business process diagrams on different levels of abstraction.

Responsibility

P. Gustiené 100%

Publication number 5 (2004): “Towards the Enterprise Engineering Approach for Information System Modelling across Organizational and Technical boundaries”

Table 10: Distinguished characteristics of publication number 5

<table>
<thead>
<tr>
<th>Authors</th>
<th>R. Gustas, P. Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research questions</td>
<td>What are the common bases for the integration of various dependencies that are used in requirements engineering and conceptual modelling at the pragmatic, semantic and syntactic level? How can graphical models at various levels of abstraction be used for creation and justification of a supporting technical system specification in the context of organisational process models? What are quality criteria for the evaluation of diagrammatic constructions for the representation of information system solutions on different levels of abstraction?</td>
</tr>
<tr>
<td>Research method</td>
<td>The method of enterprise modelling was used on three levels of enterprise engineering for a case study of e-service for ordering trips.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Most information systems engineering methodologies are restrictive in a way that a supporting technical system specification can not be justified in the context of an organisational process model. The explicit representation of business infrastructure is a necessary condition to understand orderly transformations of the work practices. Traditional methods of information systems analysis and design are based on the idea of a separation of concerns, which are represented by using various static and dynamic diagrams. They do not take into account interdependencies of various diagrams as well as the communication dependencies among organisational and technical system components. Comprehensive analysis methods across organisational and technical system boundaries are not available. Without a complete understanding of organisational architecture, stakeholders have no criteria for determining the irrelevance of information flows. It results in ambiguous, incomplete and inconsistent system solutions that cause misunderstandings and semantic problems of communication. An implementation bias of many information system modelling techniques causes problems because the same concepts are applied at analysis and design stages. Enterprise architecture is necessary to provide a basis for the organisation’s information technology planning. It is necessary to introduce a common basis for the integration of various dependencies in requirement engineering and conceptual modelling on different levels of abstraction. The quality criteria of enterprise engineering are essential for understanding various issues of the organisational fitness. Such criteria play an important role.</td>
</tr>
</tbody>
</table>
Main concepts and observations

**Main concepts:** Enterprise engineering, semantics, pragmatics, technical and organisational system components.

**Observations:** Semantic diagrams of traditional methods do not usually take into account interdependencies of various diagrams as well as the communication dependencies among organisational and technical system components. The communication gap between business managers and system developers is a big problem. These actors need models and methods for early analysis of new organisational solutions prior a new supporting IT system is introduced. It might help them to define, visualise and to access various organisational changes by using graphical approach for business process reengineering. The quality criteria of enterprise engineering are essential for understanding various issues of organisational fitness.

Contribution

The results from case study done on the existing web site for ordering trips demonstrated that service-oriented modelling enterprise models can be used as a uniform basis of reasoning about quality of representations at the pragmatic, semantic and syntactic levels. My contribution includes case study examples and an analysis of desirable qualities of IS specifications. Three-level framework was also presented for justification of how technical system components fit into the overall organisational system.

Implications

The separation of pragmatic, semantic and syntactic system models leads to a natural division of enterprise engineering products into three different representations. All three levels of models should contribute to the extended methodology of an enterprise-wide engineering technique. Enterprise models can be used as a tool for analysis of business solutions prior a new supporting IT system is introduced. Such models can be considered as a corporate resource in diagnosing potential problems, and also are crucial for reasoning about business process integrity and the purposeful implications of an organisational change. Various diagrammatic constructions that are used for representation of information system solutions on different levels of abstraction should be evaluated by certain criteria to ensure system quality. Enterprise models provide a basis for system quality control. They can identify and control the following design qualities such as ambiguity, integrity, consistency and completeness and provide a basis for gradual understanding of why and how various technical system components come about.

Responsibility

R. Gustas 60%
P. Gustiené 40%


**Table 11: Distinguished characteristics of publication number 6**

<table>
<thead>
<tr>
<th>Research question</th>
<th>R. Gustas and P. Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can pragmatic dependencies drive service-oriented information systems analysis and design process?</td>
<td>How can pragmatic aspects be mapped to conceptual representations at the semantic level, which defines semantics of business design, defined in terms of static and dynamic relationships among concepts?</td>
</tr>
<tr>
<td>How should pragmatic aspects be reached using service-oriented modelling method?</td>
<td>How can semantic integrity control between static and dynamic aspects be reached using service-oriented modelling method?</td>
</tr>
</tbody>
</table>
Research method

A case study was done on a Travel Agency example for the demonstration of interplay between the pragmatic and semantic dependencies. The example demonstrated how an analysis of pragmatic entities such as goals, problems, and opportunities helps system development experts to reason about advantages and disadvantages of business process activities. A graphical example was presented for bridging from the pragmatic level to the semantic level.

Premises for the research

The success of a service-oriented analysis and design (SOAD) approach depends on finding the appropriate fitness between the pragmatic level, which defines the overall business strategy, and the semantic level that defines business processes across organisational and technical system boundaries. Pragmatic descriptions aim to provide the motivation for conceptual representations at the semantic level, where enterprise components as service requesters and service providers are defined by interactions. The important part of the business process analysis is how different goals, opportunities and problems are related and how these relationships are guided, as well how pragmatic aspects, which define the ‘why’ issues, are further analysed and presented in static and dynamic perspectives. Pragmatic dependencies can be used to analyse the business process in terms of goals, problems and opportunities. Such dependencies can be viewed as a modelling basis to characterise intentions about new system architecture. Pragmatic descriptions should be also used for explaining information system architectures, which can be defined as compositions of service interactions. Pragmatic entities provide guidance about the incompleteness of semantic descriptions of service-oriented architectures. Comprehensive business system engineering cannot separate the pragmatic details of service usage from the semantic description of services. A holistic understanding of semantic and pragmatic dependencies between service requestors and service providers would include an analysis of service objectives and the conceptualisation of their interaction links. The analysis of pragmatic aspects is important as it drives the overall system engineering process.

Main concepts and observations

**Main concepts**: Service-oriented analysis and design, service, pragmatic dependencies, semantic dependencies.

**Observations**: Enterprise reengineering can be regarded as a process of eliminating the problematic part of IS specification and complementing it by desirable services. Semantic specification of a problem should be considered a part of a IS specification of the actual situation. On the other hand, opportunities and goals represent what is desirable in a new system. Goals can be used to specify the desirable changes that are represented at the semantic level in terms of interaction loops between actors. Opportunities may describe various alternatives for solving a specific problem and reaching a goal. If the designer would like to take advantage of a specific opportunity, then its semantic fragments must be integrated into the desired specification. IS reengineering activity should always produce a solution that satisfies a number of goals. In SOAD approach, an opportunity cannot be defined by only representing its negative influence to some problem. Problems are not completely clarified without stating a goal. Enterprise reengineering activities at the semantic level should be driven by predefined specifications at the pragmatic level. Vice versa, operations with goals, problems and opportunities should have consequences on specification fragments at the semantic level. A goal can be achieved by avoiding a problem.
or using some opportunity. It should be noted that the notion of a problem and a goal is relative. The achievement of some goal for one enterprise actor can be regarded as a problem for another actor. An analysis of conflicting pragmatic entities can be done using a hypothetical semantic difference operation, which is supposed to compute a semantic difference between two graphical descriptions.

Pragmatic dependencies can be used for reasoning about the completeness of system descriptions at the semantic level. Business process descriptions in terms of goals, problems and opportunities should be used as a driving force of enterprise modelling at this level. In conventional system development approaches, formalisation of such operation is very difficult, because the static and dynamic aspects are analysed in isolation.

The interpretation of problem and goal has some implications on how different pragmatic notions are perceived by stakeholders. In the initial phase of modelling, the goals and problems can be analysed in isolation. Nevertheless, consensus on a specific problem or goal cannot be reached without a detailed analysis of a set of semantic dependencies in the actual, and in the desired, situation. The semantics of a problem cannot be stated without complete understanding what is desired in a new system. On the other hand, the semantics of a goal cannot be specified without understanding what is undesirable in a problematic situation.

**Contribution**

A new pragmatic-driven approach for a service-oriented information systems analysis and design was presented. Its uniqueness is in the developed design foundation for graphical descriptions of the pragmatic and semantic aspects of the business processes that is based on service-oriented principles. The graphical example of the case study was presented for the demonstration of the interplay between pragmatic and semantic specifications. It was also demonstrated that evolution or change of business process fragments can be supported by using operations of semantic difference, union and intersection on goals, problems and opportunities. These operations are important as they are supposed to compute the semantic difference between two graphical descriptions. My contributions in this publication are threefold: 1. A case study was presented for the demonstration of the interplay between pragmatic and semantic specifications. 2. The analysis of pragmatic entities was done using decomposition and refinement links. It demonstrated bridging from pragmatic level to semantic specifications, which are defined in terms of interactions between service requester and service provider. It demonstrated how new pragmatic entities are integrated into the overall business process, and how these entities are integrated into the existing enterprise model at the semantic level. 3. Presentation of meta-model to show different possible interpretations of concepts used in service-oriented modelling method.

**Implications**

Semantic integrity control between static and dynamic dependencies of business processes is one of the major benefits of the service-oriented analysis and design process. It is driven by pragmatic descriptions, which are defined in terms of pragmatic dependencies such as goals, problems and opportunities, negative and positive influences and refinement. Pragmatic descriptions aim to provide an explanation for conceptual representations of enterprise components in terms of services, which are defined by interaction dependencies between service requesters and service providers. The underlying assumption is that IS services are worthwhile if they meet the goals of the organisation. Goals can be used for the semantic completeness control or for the identification of new business process boundaries. The SOAD approach
put emphasis on computation-neutral modelling of business processes in terms of services. Pragmatic specifications that are business-oriented are useful for two reasons; they justify service events and can be seen as a driving force in service-oriented reengineering of business process patterns.

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>R. Gustas 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P. Gustiené 50%</td>
</tr>
</tbody>
</table>

Publication number 7 (2009); “Service-Oriented Foundation and Analysis Patterns for Conceptual Modelling of Information Systems”

Table 12: Distinguished characteristics of publication number 7

<table>
<thead>
<tr>
<th>Authors</th>
<th>R. Gustas and P. Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>What are the fundamentals of a service-oriented method for conceptual modelling of information systems? How can the concept of service be used for the integration of static and dynamic aspects into one modelling notation? How could service-oriented method be applied for analysis patterns for the conceptual modelling of information systems?</td>
</tr>
<tr>
<td>Research method</td>
<td>Ontological perspectives were applied for ontological definition of enterprise system in a service-oriented foundation. A service-oriented foundation was applied to construct analysis patterns for conceptual modelling of information systems.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>One of the reasons why the traditional methods do not provide effective support for engineering information systems is that service architectures are difficult to visualise and comprehend for business experts who really define the organisational strategy. The concept of service is not explicitly used in traditional methods for analysis and design of information systems. Traditional methods cover only a part of the required modelling notations that are currently emerging under the service-oriented analysis and design approaches. There is a lack of an integrated method for modelling service architectures and the lack of principles for the interplay between static and dynamic models, which are crucial for gluing the strategic, organisational and technical descriptions into a single, computation-neutral and integrated representation. Interteldependencies among models and perspectives that specify information systems can not be analysed in isolation, because they define the same artefact. Communication action has two orthogonal aspects, the cohesion of which can allow the representation of static and dynamic aspects into a single modelling notation. Traditional information systems modelling notations do not combine static and dynamic aspects, but represent those using different types of diagrams. Service-oriented foundation can be used for constructing analysis patterns for service-oriented modelling.</td>
</tr>
<tr>
<td>Main concepts and observations</td>
<td>Main concepts: Service, service architecture, ontology, intersubjective and objective perspectives of communication action. Observations: The changes of service architecture need to be constantly captured, visualised and agreed upon. One of the reasons why the traditional methods do not provide such support is that service architectures are difficult to visualise and comprehend for business experts who determine the organisational strategies. The concept of service and service-oriented approach could help to solve that problem. The most fascinating idea about the service concept is that it applies equally well to organisational as well as technical components, which can be viewed as service requestors and service providers.</td>
</tr>
</tbody>
</table>
The primary goal of service-oriented architecture is to align the business process models with the information system design in order to make both organisational and technical system parts more effective. The main goal of this paper is to present a new service-oriented foundation and the associated analysis patterns for computation-neutral information system modelling. The service-oriented approach was applied for the integration of static and dynamic aspects into a single diagram.

Contribution
One of the main contributions of this research study is presentation of a service-oriented foundation that is based on the interplay between two orthogonal aspects of communication action: intersubjective and objective. These two perspectives facilitate the reasoning and understanding of different components and service compositions. It was demonstrated that the interplay between these two perspectives enables integration of static and dynamic aspects using a single modelling notation.

The application of the service-oriented foundation for the construction of analysis patterns such as sequence, selection, synchronisation, iteration and search is one of my main contributions in this research study. My contribution is the adoption of our enterprise modelling approach for service-oriented analysis. I have introduced five mandatory attribute dependencies and a composition dependency for the unambiguous service-oriented modelling of structural and dynamic aspects of system specifications.

Implications
By using the service-oriented approach, the ontology of enterprise system can be represented on different levels of abstraction and it can be bridged to various perspectives. Therefore, service architecture specifications can be used as a basis for reasoning about semantic completeness, consistency and continuity of information systems design. The presented graphical approach opens a totally new way of enterprise system engineering from services that span across the organisational and technical system boundaries. The self-describing nature of interaction loops, and particularly the ability to define new configurations of service architectures, provides significant competitive advantages. Adopting a new service-oriented paradigm has the potential to manage information systems development complexity and to lower enterprise architecture maintenance costs. A new way of modelling provides improved integrity of service architectures across different perspectives and along various dimensions.

Responsibility
R. Gustas 60%
P. Gustiené 40%

Table 13: Distinguished characteristics of publication number 8

<table>
<thead>
<tr>
<th>Research question</th>
<th>P. Gustiené and R. Gustas</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can the service-oriented paradigm contribute to the control of business process continuity and integrity in IS analysis and design?</td>
<td></td>
</tr>
<tr>
<td>How can the business process be conceptualised using basic service-oriented constructs?</td>
<td></td>
</tr>
<tr>
<td>How to map conceptual representations of services to components and object-oriented diagrams?</td>
<td></td>
</tr>
</tbody>
</table>

**Research method**
The service-oriented foundation was applied for the demonstration of interplay of three basic constructs by one interaction loop and illustrated using an example of Recruitment Management Service. The demonstration bridged from conceptual representation of services to components, as well as from a service-oriented diagram to object-oriented diagrams.

**Premises for the research**
The conventional methods for IS analysis and design are not based on a service-oriented paradigm that facilitates control of business process continuity and integrity. Traditional information system modelling methods covers just a part of the required modelling notations that are currently emerging under the service-oriented analysis and design (SOAD) approaches. They do not put into the foreground the service-oriented principles that are crucial to glue artefacts of various diagrammatic types. The lack of research on semantic integrity among different types of diagrams is not a new issue. The principles of the integration of static and dynamic aspects as well as the principles of the separation of concerns are not clear in the conventional system analysis and design methodologies. The analysis of static and dynamic parts in isolation results in the need for graphical representations for the semantic consistency and integrity control across various dimensions. The integration of internal and external behaviour which is encapsulated in a service concept provides modelling flexibility and enables the integration of static and dynamic aspects of service architecture.

**Main concepts and observations**
**Main concepts:** Service-oriented analysis and design, service-oriented constructs and basic events.  
**Observations:** Service-oriented analysis and design is a new approach that has evolved from object-oriented and component-based software engineering. Many approaches are focusing on the design of services from software components; such focus is not directly applicable for conceptual modelling of services. Service-oriented architecture (SOA) provides a set of design principles that can be applied for an analysis of business processes as services. An enterprise system can be defined as a set of loosely-coupled interacting components (organisational or technical), which are able to perform specific services on request. Service-oriented constructs are based on three types of events: creation, termination and reclassification. The composition of these three types of basic constructs can be used for the conceptualisation of intersubjective and objective perspectives of service architecture, which is important for conceptualisation of objects in a service interaction loop.

**Contribution**
It was demonstrated how conceptual representations of service architecture can be presented using one or more interaction loops. The semantics of one interaction loop can be defined by using any two basic constructs. The interplay of three basic constructs was illustrated with the modelling example. My contribution in this research study was the demonstration of bridging from conceptual representations of service architecture to component-based representation. The semantics of service-oriented artefacts that are computation-neutral were analysed in terms of their associated counterparts used in object-oriented diagrams. It was shown how service-oriented constructs predefine the semantic details that can be used for the elicitation of objects-oriented operations.

**Implications**
Conceptual representations of service architectures define computation independent aspects of business processes, which are not influenced by the implementation-dependent solutions. The semantics of service-oriented events
can be explained in object-oriented design terms. The events presented with UML notation creates difficulties for validation of the diagrammatic presentations of solutions used by business process analysis experts. Such diagrams are prone to inconsistencies, discontinuity and ambiguity problems. Service-oriented constructs are quite comprehensible and can be communicated among business experts and designers more effectively than a set of various types of object-oriented diagrams. The service-oriented modelling approach is aiming at an engineering process that is based on a single model, which is used to conceptualise service architecture. It enables integration of static and dynamic aspects. Computation-neutral notation demonstrates that service-oriented constructs are simpler and could be more comprehensible for business people. It can better contribute to bridging the communication gap between system designers and business experts.

Responsibility

P. Gustiené 60%
R. Gustas 40%


Table 14: Distinguished characteristics of publication number 9

<table>
<thead>
<tr>
<th>Authors</th>
<th>R. Gustas and P. Gustiené</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How traditional methods for conceptual modelling represent intersubjective and objective perspectives of system specifications and how can these perspectives, which define static and dynamic aspects, be interpreted using a service-oriented method? How can event/action rules that specify objective aspects be defined using a service-oriented method?</td>
</tr>
<tr>
<td>Research method</td>
<td>Two significant qualities that characterise information system development traditions, namely intersubjective and objective were discussed and presented. It was explained how these traditions are presented in an object-oriented approach. A comparison was made between service-oriented modelling and object-oriented modelling. A graphical presentation was done for showing the way to integrate static and dynamic aspects. It was explained and presented using a simple Switch on Clock example.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Service architectures are intrinsically complex engineering products that can be defined on different levels of abstraction and represented using several dimensions. One of the reasons why traditional methods do not provide effective support is that service architectures are difficult to visualise across disparate modelling dimensions. Another problem is that the same implementation-dependent techniques are used both for system analysis and design. The concept of service is not explicitly used in the conventional information system methodologies, in spite of the fact that this concept is rather well understood in different domains and can be equally applied for conceptualisation of organisational and technical system components. Information systems can be structurally visualised as evolving conceptualisations of service architectures. Service-orientation promotes autonomy, flexibility and interoperability of enterprise system components. The changes of enterprise architecture (EA) can be enabled by the creation, removal or replacement of loosely-coupled information system components. Traditionally, graphical representations of EA are built fragment by fragment</td>
</tr>
</tbody>
</table>
and, when all is done, the business design is typically not aligned with the information system design. It is very difficult to maintain semantic integrity of multiple EA specification fragments. Service-oriented models represent only computation-neutral aspects. Such graphical representations are less complex and can be successfully used by non-technicians who play a key role in system integration. It is recognised that object-oriented approach and UML support for such tasks is quite vague, because semantic integration principles of different diagram types are still lacking. A mechanism of semantic integrity control of static and dynamic aspects is not clear in object-oriented approach.

<table>
<thead>
<tr>
<th>Main concepts and observations</th>
<th>Main concepts: Service orientation, service architecture, intersubjective and objective perspectives, static and dynamic aspects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
<td>Service architecture is not necessarily bound to the technical aspects of information system development. It can be defined by using conceptual models that are independent of any implementation technology. Unfortunately, the conventional information systems analysis and design methods cover only a part of the required modelling notations for the engineering of service architectures. Service-orientation is a paradigm that can be applied for the conceptual modelling of information systems. The concept of service is rather well understood in different domains. It can be applied equally well for conceptualisation of organisational and technical information system components. This paper concentrates on an analysis of the differences between the service-oriented modelling method and object-oriented modelling approach, concerning intersubjective and objective perspectives.</td>
</tr>
</tbody>
</table>

| Contribution                  | The paper presents a comparison of two perspectives in the service-oriented modelling method and the object-oriented modelling approach. My contribution in this publication is the comparison of intersubjective and objective perspectives in both modelling methods. It was demonstrated how information system architecture can be conceptualised using service-oriented modelling method. It was concluded that by matching the actor dependencies, one can explore responsibilities that are predefined to different actors. The Clock example was used to demonstrate service interactions, which enable interplay between intersubjective and objective perspectives. Cohesion of these perspectives is crucial to maintaining a holistic representation, where external and internal views are visualised together. |

| Implications                  | Understanding enterprise system architecture depends on knowing how different subsystems are interconnected. Interactions among enterprise system components are used to conceptualise the semantics of information system data and process. The interplay between intersubjective and objective perspectives using one service-oriented diagram facilitates a better semantic integrity control between static and dynamic aspects. A new service-oriented method for system analysis and design should bring significant benefits including improved ability for organisations to maintain strategic knowledge in a systematic way, and reduced costs for a systematic analysis of new IT solutions before they are implemented. It also enables the improvement of integrity and traceability of knowledge within companies by providing comprehensible service architecture descriptions. |

| Responsibility                | R. Gustas 60%  
P. Gustiené 40% |

Table 15: Distinguished characteristics of publication number 10

<table>
<thead>
<tr>
<th>Authors</th>
<th>P. Gustiené, I. Peltomaa, H. Helaakoski</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How can Sebi-framework, used for information integration using semantic technologies, be extended by a service-oriented approach to ensure a better quality of semantic interoperability?</td>
</tr>
<tr>
<td>Research method</td>
<td>Analysis of Sebi-framework and SOAD approaches.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Information systems have an increasingly important role in the realisation of business process demands, which leads to a demand of close interaction and understanding between organisational and technical components. It is critical for enterprise interoperability, where semantic integration of information and technology is the prerequisite for successful collaboration. Growing business enables growing data, and companies will suffer service disruptions if there is no strategy for how to manage relevant information. Semantic problems of communication between business analysis and design experts lead to ambiguous and incomplete system specifications as well as it causing enterprise interoperability problems. The success of enterprise interoperability greatly depends on how the static and dynamic aspects of the enterprise are integrated. The description of integrated service architectures should be established before implementation-specific solutions are discussed. Business solutions presented by integrated conceptual modelling techniques provide guidelines for implementation issues. They provide justification and fitness rules that are important to link implementation-independent and implementation-dependent modelling levels.</td>
</tr>
</tbody>
</table>

Main concepts and observations

Main concepts: Semantic framework, information integration, service-oriented analysis and design,

Observations: Semantic interoperability can be ensured by providing contextual knowledge of domain applications. Interoperability is comprised by both technical and information integration. The main technical challenge is the lack of interoperability of different systems and data sources. The problem is that most attention is placed only on technical integration solutions. Information integration is enabled by semantic interoperability that focuses on data inside the enterprise as well as focusing on how to interoperate software systems outside the origin enterprise. Sebi framework uses semantic technologies to enable information sharing among separate information systems. The framework provides four sub-processes: Case Envisioning, Business, Expertise and IT Domains. In order to succeed in the integration ontology development, the communication gap between business and IT-experts has to be eliminated. The service-oriented modelling approach for system analysis and design provides an integrated method and techniques that enables integration of static and dynamic aspects and provides possibilities to check consistency and completeness, and to trace undesirable system qualities. It could lead to a better quality of IS specifications that lead to a reduction of semantic problems of communication between stakeholders.
The main contribution of this paper is the construction of a new semantic framework that extends the existing Sebi-framework with a service-oriented approach for IS analysis and design.

The service-oriented approach presented for analysis and design process (SOAD) has the semantic power to conceptualise organisational and technical system components, by distinguishing intersubjective and objective views that facilitate better semantic integrity control between the static and dynamic aspects. The advantage of such modelling is that it integrates the semantics of different aspects into one type of diagram. The conceptual representation of service architectures defines computation-neutral aspects that are not influenced by any implementation solutions and are more comprehensible for business people as well as system designers. The SOAD approach provides modelling techniques to overcome such undesirable system characteristics as inconsistency, incompleteness, incoherence, redundancy and ambiguity of data. It will contribute to an analysis and elicitation only of relevant enterprise data. This extension can contribute to a better ontology integration development, which is the most important part of the Sebi-framework. The presented semantic framework is important for maintaining a holistic representation of the enterprise, which is necessary for systematic analysis of service architectures as well as of enterprise interoperability where internal and external views are visualised together.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Prima Gustiènè, Sten Carlsson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How do different fundamental theoretical linguistic ideas influence the assumptions concerning understanding and communication? How do different fundamental ideas for learning influence an understanding of models? How can a service-oriented modelling method facilitate improved communication and learning?</td>
</tr>
<tr>
<td>Research method</td>
<td>Analysis and literature review of fundamental theoretical ideas concerning, language, communication and learning. An analysis of the problems in existing modelling approaches, as far as theoretical assumptions concerning understanding, language and communications are concerned, was presented. Identification of theoretical assumptions necessary to take into account while creating graphical representations and methods for information system analysis and design. The service-oriented modelling method was presented and promoted as a possible solution for improving modelling methods for information systems.</td>
</tr>
</tbody>
</table>
| Premises for the research | Various models and methods are used to support the information systems development process, but after many years of practice, information systems development projects still continue to fail. The questions we raise here are: is

there something wrong with the models and methods for information systems development concerning learning and communication? Are they not good enough to describe the enterprise services? Are they not good enough as a means of communication among stakeholders involved? One of the reasons is that the conventional modelling approaches do not provide sufficient support for learning and communication among stakeholders. The models and methods used as a means of communication should have a reasonable pedagogical capacity that promotes an understanding of information system contents and communication among all the partners involved. The objective of this research is to present a comprehensive review of fundamental theoretical assumptions concerning communication, understanding and learning as well as to present the basic problems in traditional modelling approaches for information systems analysis and design and provide the assumptions that should be taken into consideration while constructing models and methods to support communication and learning process.

| Main concepts and observations | **Main concepts**: Learning, communication, modelling, information system analysis and design, intersubjective and objective perspective.  
**Observations**: One of the aims of the models used during the information systems development process is to promote understanding among stakeholders as well as to have a reasonable pedagogical capacity in the learning process. Mutual understanding and agreement among stakeholders is important for successful communication. Semantic problems of communication are very often causes of failures of IS development projects. A clear understanding of the notion of communication, learning and understanding and the relationship between them, is critical when constructing models and methods for information systems analysis and design. It is necessary that the theoretical grounds of communication and learning should be taken into account and integrated in the system development process.  
Graphical representations used during the information systems development process should promote understanding among stakeholders involved. They should have enough semantic power to represent what is necessary to represent, as well as having reasonable pedagogical capacity in the learning process. As models and methods are made by people and first of all for people, they should be based on the foundations that support communication, understanding and learning. They should be based on theoretical foundations that facilitate the understanding of complex content of information systems. |
Contribution

The main contribution of this paper is a comprehensive review of fundamental theoretical assumptions concerning communication and learning, as well as the statement of some basic problems in traditional modelling approaches for analysis and design of information systems in this perspective. The outcome of this theoretical review suggested some theoretical grounds, which lie in the foundation for the service-oriented modelling method. The analysis made it clear that intersubjectivity and mutual understanding among stakeholders is one of the critical points in the IS development process. The models should be able to represent both the parts and the whole according to a hermeneutical circle.

The service-oriented method was shortly presented as a possible solution. The advantages of the service-oriented modelling method are as follows:

• service orientation combines intersubjective and objective perspectives into one modelling notation, which facilitates understanding and reasoning about service architectures across organisational and technical system boundaries. Using just one model facilitates better learning.
• service-oriented analysis is computation-neutral. A service-oriented way of thinking is based on the foundations of communication. This way is more comprehensible from a pedagogical point of view and facilitates the learning process. It also facilitates the involvement of stakeholders without deep technical knowledge in the area of information systems and can therefore contribute to bridge the gap between enterprise business architects and system designers.

Implications

Being pragmatic-driven and computation-neutral, the service-oriented modelling method supports systematic way of analysis, starting from business-oriented issues to structural and dynamic aspects. By having the possibility to integrate structural and dynamic aspects, the method helps to achieve semantic integrity between different modelling dimensions and to reduce the communication gap among stakeholders. Such an integrated way of modelling has a reasonable pedagogical capacity, and facilitates human understanding and learning.

Responsibility

P. Gustiené 70%
S. Carlsson 30%

6.2. Summary of the Research Results Presented in the Publications

The contribution of 11 publications is summarized in this section. It is presented together with the motivation behind the research studies. Each publication contributed to the development of the service-oriented modelling method.

Publication 1, “On a Problem of Ambiguity and Semantic Role Relativity in Conceptual Modelling” (2002), describes a common problem of semantic ambiguity in the area of conceptual modelling. The research contributes with an identification of the relativity of semantic roles. The problem of semantic role relativity is also relevant for the conventional approaches of information system analysis and design. The paper argues that it is impossible to set up a strict
classification of semantic roles. The strict classification of semantic roles hides important semantic details about a problem domain. It is a cause of structurally different, but semantically equivalent conceptual representations of enterprise architecture fragments. An analysis of the semantic roles of the noun phrases, as well as the role played by each noun phrase in relation to the verb, and the detection of the relativity of semantic roles, leads us to understand that the interpretation of concept roles in service-oriented way of modelling should be flexible. The major conclusion is that the interpretation of concept roles cannot be strict, since it depends on semantic relations to other concepts. The inability of contextual relation control in conceptual modelling causes various problems in the area of view integration, natural language processing and information systems analysis. A service-oriented foundation can be used to solve the role relativity problem in a more flexible way, since as service-oriented way of modelling is based on a flexible interpretation of roles.

Relativity of semantic roles contributed to the identification of one of the most important characteristics of service-oriented modelling method that interpretation of the concept depends upon the types of semantic links through which it is related to other concepts. Whether a concept is regarded as an instance, class, flow, state, pragmatic entity or an actor depends upon types of the semantic links it is related to other concepts. The research studies on semantic role relativity also contributed to the identification and definition of active and passive concepts of service architectures. This distinction is important for two reasons:

1. the identification of active objects enables the separation of the intersubjective perspective, which is important for an analysis of external actors, their structure and responsibilities.
2. the identification of passive objects is important for an analysis of internal structural changes, which helps to separate objective perspective. It helps to understand behavioural aspects of business data.

An interplay between these two perspectives enables the integration of the static and dynamic aspects of service architectures.

The major effort of the research done and the results presented in Publication 2, “Extending Lyee Methodology using the Enterprise Modelling Approach” (2002), is the demonstration how the Lyee software engineering methodology can be extended by using a set of semantic constructs in combination with the pragmatic-driven approach for IS analysis and design. One of the objectives of this research was the adoption of the graphical models that facilitate the detection of semantic problems of IS and software specifications. The modelling method, which is presented in this thesis, was used as a uniform basis of reasoning about ambiguity, coherence and completeness of processes and data that are defined on the neighbouring levels of abstraction. The results of the research contributed with the knowledge of incremental design, which motivates how various layouts, computation rules and process route diagrams are justified by enterprise architecture. It provided a basis for the gradual understanding of how pragmatic, semantic and logical software requirements can be combined and integrated. The analysis of pragmatic specifications is important because they define the aspects of enterprise architecture,
which are defined at the semantic level. Conceptual representations that are defined at the semantic level justify logical software requirements expressed at the syntactic level. In such a way, service-oriented conceptualizations can be used for the verification of implementation-specific requirements. On the other hand, the graphical representations of enterprise architectures can be validated by the requirements presented at the pragmatic level. The research idea is based on the assumption that the intentions of business experts are represented by a set of pragmatic dependencies, which drive the overall information systems engineering process. The results of this paper contributed to the development of pragmatic-driven approach for service-oriented analysis and design. In the Lyee project, pragmatic and semantic specifications were used as a basis for the detection of inconsistencies, ambiguities and incompleteness of implementation-specific diagrams. The research results of this project contributed with knowledge for developing a systematic modelling approach to software engineering. This way of modelling is critical for understanding the fitness between organisational and technical system parts.

Publication 3, “On Desirable Qualities of Information System Specifications” (2003), contributes with knowledge of how to identify and confront undesirable characteristics in enterprise architecture conceptualisations. The detection of undesirable qualities of diagrams helps system designers focus on the improvement of the diagram quality. Semantic quality is essential when the enterprise engineering product is intended to enable effective communication of various architectural solutions among different stakeholders involved in the IS development process. An integrated modelling method of dealing with the graphical dependencies at various levels of abstractions helps in using service-oriented diagrams for the identification of such undesirable semantic qualities as inconsistency, incompleteness, incoherence, ambiguity and redundancy. Ambiguities of concepts indicate a lower semantic quality of the system conceptualisation. Ambiguities can be eliminated using a set of the mandatory attribute dependencies that is presented in the theses. The identification of the noteworthy difference among a set of the mandatory attributes of every concept is a very important technique. It contributes to a better quality of system specifications and to improved communication among stakeholders, since every concept can be distinguished by its unique similarities and differences. The detection of two undesirable characteristics such as ambiguity and redundancy, using the inference rules of static dependencies, was presented. Rules are focused on identifying a redundant inheritance dependency and redundant mandatory attributes (for different classes) in the objective perspective. A service-oriented analysis of the business process and data makes it possible to study the conceptual models in the context of interaction loops or in connection to the different message flows the concepts are related to. An analysis of the semantic dependencies between concepts, in combination with inference rules, is one possible way for the identification of undesirable characteristics of graphical representations. The presented approach contributed with the evaluation techniques for understanding and detecting undesirable semantic qualities of service-oriented diagrams.
The presented research results of Publication 4, “Enterprise Modelling Approach for Information System Engineering” (2003), are results of the project of the HumanIT program at Karlstad University. The HumanIT research program is a multidisciplinary attempt to focus on the human value of IT. To obtain value from information technology resources, it is necessary to understand the environment were IT resources are used. The major focus of the research studies was on the demonstration of how modelling techniques can be used to bridge organisational and technical system specifications. Organisational changes must be consistent with the newly introduced technical solutions that support these changes. Consistency between organisational and technical changes helps to understand human value of IT. The presented research demonstrated the way in which various organisational and software architectures are bridged. The interactions of enterprise system components must be structurally and behaviourally coherent with respect to other elements of the global enterprise model. The paper presented the ways that help to systematically identify semantic conflicts, which result in communication failures during the enterprise modelling process. The method is centred around the idea of a model-driven approach, which focuses on early modelling of IS development that concentrates on what the desired system must do, but not how. The results of this research contributed with a model-driven and computation-neutral approach for IS analysis and design that is more understandable for people with little or no technical background.

Publication 5, “Towards the Enterprise Engineering Approach for Information System Modelling across Organizational and Technical Boundaries” (2003), revisits various aspects of enterprise system engineering on the pragmatic, semantic and syntactic levels of abstraction. The paper highlights that most of the existing methodologies put emphasis on the computerised part of IS design. One of the major problems is that comprehensible computation-neutral methods across organisational and technical system boundaries are not available. The main contribution of this paper is the framework that describes three different modelling techniques that are necessary to provide alignment of computation-neutral descriptions with various aspects of IT-system design. The presented approach helps to detect the defects of organisational and technical fitness, to understand why technical system components are useful, and how they fit into the service loops of the overall organisational system. For the achievement of these goals, three modelling levels, pragmatic, semantic and syntactic are introduced. The proposed modelling techniques were used to demonstrate IS engineering approach and the usage of three different types of specifications in the context of IS analysis and design. The case study example demonstrated that enterprise models are relevant for reasoning about business process semantic integrity and the pragmatic completeness of IS architecture. The presented modelling techniques suggest a new systematic approach for bridging integrated business process and data descriptions with implementation-oriented IS specifications. It can be used to justify such logical elements as screen forms, database files and software components. The presented techniques help to improve the overall quality of enterprise engineering across organisational and
technical system boundaries. Undesirable qualities of conceptualisations, such as semantic inconsistency, semantic incoherence, semantic redundancy, semantic incompleteness and ambiguity, can be identified with the help of discrepancies among diagrams, which are presented on three levels of abstraction. The main contribution of this study is a theoretical motivation of service-oriented modelling method. It presented the architectural framework and modelling techniques on three levels of abstraction.

Publication 6, “Pragmatic-Driven Approach for Service-Oriented Analysis and Design” (2008) emphasises the importance of computation-neutral modelling and bridging between the pragmatic and semantic aspects of business processes and data. The research on the relativity of semantic roles (Publication 1) has also contributed to the development of a pragmatic-driven, service-oriented analysis and design method. It contributed to the presentation of a meta-model to show different possibilities of concept interpretation in the service-oriented modelling method. Pragmatic aspects of services can be defined in terms of goals, problems and opportunities, which are relative notions. Whether a situation is desirable or undesirable depends on the designer's goals. Information systems analysis and design can be regarded as the process of eliminating the problematic part of specification and complementing it by desirable service architecture. The semantic specification of a problem should be considered as part of a specification of the actual situation. On the other hand, opportunities and goals represent what is desirable in a new system. Goals can be used to specify the desirable changes that are represented at the semantic level in terms of interaction loops between actors. Opportunities may describe various alternatives for solving a specific problem and reaching a goal. If the designer would like to use a specific opportunity, then its semantic fragments must be integrated into the new specification. A system reengineering activity should always produce a solution that satisfies a designer goal. In the pragmatic-driven approach, the opportunities are expressed by using the negative and positive influence dependencies with respect to designer's problems and goals. The desired situation of services, which are expressed in terms of goals, cannot be clarified without taking into consideration existing problems.

This study on service-oriented analysis and design is focused on pragmatic descriptions of stakeholders, which can be interpreted as a driving force of the overall system engineering process. The intentions of stakeholders are graphically described using a set of pragmatic dependencies. It was demonstrated that pragmatic entities such as goals, can be used for the semantic completeness control of service-oriented diagrams. Change management and the evolution of business process fragments can be supported by using operations of semantic difference, union and intersection on goals, problems and opportunities. Pragmatic dependencies were used for the analysis of actor intentions, which enable the identification of possible contradictions between desirable and undesirable situations. It was demonstrated that goal hierarchies can be used to identify business process boundaries and enable reasoning about semantic completeness of service architectures. In the presented approach, the semantic fragments of computation-neutral specifications are defined using service interaction
loops. Interaction dependencies specify services in terms of communication actions, which play an integral role in binding the static and dynamic aspects.

The Travel Agency case study was presented for the demonstration of bridging from the pragmatic level to semantic specifications. The interplay between the pragmatic and semantic descriptions was demonstrated, which provides determination for technical and organisational system components. In other words, pragmatic specification drives the overall service-oriented analysis and design process, which is one of the benefits of new method. The presented examples demonstrate how goals, problems and opportunities help system analysis experts to study advantages (positive influence link) and disadvantages (negative influence dependency) of business interactions. Pragmatic specifications in terms of goals, problems and opportunities drive the overall service-oriented analysis and design process. It is one of the benefits of service-oriented modelling method.

The goal decomposition and goal refinement links help IS designers to understand enterprise architecture in terms of service loops. It is not enough to define every sub-goal in terms of interaction dependencies. Service interactions must be integrated using behavioural and structural descriptions of services. Pragmatic hierarchies provide much of the desired knowledge for the detection and elimination of the problematic fragments of specifications at the semantic level and complementing them with the desirable fragments. Decomposition hierarchy drives the strategies and ways in which the high-level objectives are achieved. That is why, at the bottom level of the hierarchy, it is necessary to refine pragmatic notions in terms of interactions between service requesters and service providers. Such a systematic way of modelling, bridging from the pragmatic to semantic level, ensures the desired requirements traceability (Maciaszek, 2005; Pohl, 1996). It enables tracking changes from or to a requirement during the development process. It is essential for developing and maintaining high quality software and is critically important for systematic change management. The pragmatic-driven approach for service-oriented analysis and design is one of the most important research results presented in this thesis.

Publication 7, *“Service-Oriented Foundation and Analysis Patterns for Conceptual Modelling of Information Systems”* (2009), contributes with the definition of three basic semantic constructs, which are expressed using creation, termination and reclassification events. A new service-oriented modelling foundation is based on the integration of intersubjective and objective information system analysis and design traditions. The interplay between basic constructs enables this integration. Continuity of interaction loops is the main modelling characteristic of service-orientation. Enterprise system architecture can be defined as a set of interacting loosely connected organisational and technical components, which are able to perform specific services on request. Cohesion of the intersubjective and objective perspectives of interaction loops, allows the representation of static and behavioural aspects using a single modelling notation. An advantage of this new principle is that it allows controlling the semantic integrity of static and dynamic aspects in service-oriented diagrams. The presented foundation is fundamental to the developing of system analysis patterns, which can be used for a service-oriented
modelling of information systems. The presented set of semantic dependencies is sufficient for defining the main workflow patterns. This paper defines five main service-oriented system analysis patterns. It was demonstrated how the main workflow patterns such as sequence, selection (choice and merge), synchronisation (split and join) and iteration were constructed. Various combinations of the static and dynamic dependencies used in service-oriented constructs are able to express the main workflow control patterns. Typically, only dynamic models are used in the pattern research. These modelling techniques do not take into account the structural data dimension. Separation of the static and dynamic aspects in pattern modelling creates fundamental difficulties for two major reasons. Since the static details must somehow be compensated for by using dynamic constructs, the number of the basic patterns is larger than is really necessary. If static aspects are not taken into account, then patterns will become more complicated to use for business process change management and evolution.

The research studies presented in papers 8 and 9 contribute to the research on two modelling approaches: service-oriented modelling and object-oriented modelling. The research study presented in Publication 8 demonstrates the interplay between service-oriented modelling, which is computation-neutral and object-oriented design, as well as representing services to components. This research contributes to the transition principles from service-oriented modelling to implementation-specific design. The main research goal of the research presented in Publication 8 "Introducing Service-Orientation into System Analysis and Design" (2008) was to introduce the basic semantic constructs that can be used for service-oriented analysis and the semantic integration of various modelling dimensions. The main constructs are based just on two basic events: creation and termination. These two events are fundamental to the definition of the reclassification event, which can be understood as interaction between service requester and service provider. The effect of every action is defined by using creation, termination or reclassification constructs. The composition of these three types of basic constructs into service loops is used for the conceptualisation of a continuous or finite lifecycle of one or more classes of objects. By matching the dependencies between actors (service requester and service provider), one can explore opportunities as well as responsibilities that are available to the actors. Conceptual representations of service architecture can be defined using one or more interaction loops. The semantics of one loop can be defined using any two basic constructs. The interplay between the three constructs was presented by one interaction loop as an example of an exclusive choice pattern. This research also contributes with a way of bridging from conceptual representations of service architecture to component-based representation as well as bridging from service-oriented to object-oriented diagrams. Service architecture can be implemented as a set of loosely-coupled enterprise system components. The organisational system is typically supported by a technical system part, which can be conceptualised in terms of software or hardware components. Typically, a coherent set of interactions is delegated to one independent software or hardware component. These interactions must fit together for the achievement of a common goal. Service-oriented diagrams
are composed of creation, termination or reclassification constructs. The constructs are combined for a graphical representation of the semantic details of service architectures. Traditionally, object-oriented diagrams are used in IS analysis as well as in the design phases. It makes it difficult for developers to identify noteworthy semantic differences among various states of objects. Our experience in analysing system specifications by using a computation-neutral way of modelling demonstrates that service-oriented events are more comprehensible than a set of object-oriented diagrams. Such diagrams create difficulties in the validation of graphical representations by business experts.

The research results presented in the Publication 9, “A New Method for Conceptual Modelling of Information Systems” (2009), concentrates on the analysis of differences between service-oriented modelling and object-oriented modelling. The paper presents the linkage between intersubjective and objective traditions in IS analysis using object-oriented design and service-oriented modelling.

The main advantage of service-oriented constructs is that they are easier for the verification of semantic integrity between static and dynamic aspects compared to object-oriented diagrams. Service-oriented diagrams are computation-neutral representations. They are comprehensible for business process analysis experts as well for technical system designers. This is a major reason why these diagrams can be used for bridging the communication gap between information system designers and business architects. In addition, the service-oriented diagrams are more likely to express the semantic details in more compact way. UML has become the de facto standard for system modelling. Therefore, the paper presents and compares service-oriented events with special emphasis on state-transition diagrams and object flow diagrams. The semantics of state changes are compared with creation, termination and reclassification actions. The superimposition of the static and dynamic dependencies of the service-oriented method defines the compositions of the basic event constructs, which are important to understanding the semantic details of service architectures. Structural aspects of data, together with transition dependencies between classes, define the objective perspective. It is complimentary to interactions, which define the intersubjective perspective of service architectures. The interplay between various types of diagrams is expressed using a single type of service-oriented diagram.

This paper demonstrated that service-oriented diagrams are capable of capturing event-condition-action (ECA) rules. Such rules are often used for the representation of triggering effects on transition links in the UML state diagram, but the effects resulting from actions, which are related to the static aspects, cannot be captured by the ECA triggering rules. Nevertheless, modelling of such effects is crucial for a complete definition of the interaction loop dynamics. That is why the consequences of actions are difficult to trace using object-oriented approaches. The difficulty lies in the failure to recognise the noteworthy changes that are caused by various events. This makes overall enterprise architecture prone to mistakes, which are introduced by inconsistencies, discontinuities and ambiguities. Since service-oriented diagrams provide the integration of the static and dynamic aspects, they do not suffer from the
above-mentioned drawbacks. Service-oriented diagrams are able to describe not only actions, but also conditions on classes of objects, which must hold for triggering other actions. Very often, state diagrams tend to artificially add complexity by multiplying a number of states, which cannot be justified from the business modelling point of view. On the other hand, the semantics of service-oriented constructs are comprehensible for business experts, because service-oriented diagrams deal only with computation-neutral states, which have no implementation bias. Service-oriented constructs define events that are understandable for stakeholders without technical background.

Publication 10, “Semantic Framework for Information Integration using Service-Oriented Analysis and Design” (2009), contributes with a new framework for the semantic integration of business data. A proposed framework is an extension of the Sebi-framework (Peltomaa et al., 2008), which is used in industry, with a service-oriented modelling method. The theoretical framework was developed in collaboration with the Finnish researchers from VTT Technical Research Centre of Finland. The extended semantic framework using service-oriented analysis and design is supposed to maintain a holistic representation of the enterprise. The success of enterprise interoperability depends on how enterprise business data and processes are integrated. It deals with both technical and information integration. The service-oriented modelling method has the semantic power to conceptualise organisational and technical system components, by distinguishing the intersubjective and objective perspectives, which facilitates the integration of the static and dynamic aspects. This method provides a way of modelling to identify and control undesirable semantic characteristics such as inconsistency and incompleteness of data. Taking into account all the advantages of this method, a new semantic framework was constructed. Such a framework is necessary for enterprise interoperability, where internal and external views are visualised together. A practical application of this framework is one of the intended directions of our future research collaboration.

The research results presented in the Publication 11, “How Models and Methods for Analysis and Design of Information Systems can be Improved to Better Support Communication and Learning” (2009), contributes with a literature survey on theoretical assumptions concerning understanding, communication and learning in the context of the traditional modelling approaches for information system analysis and design. This study provides the theoretical assumptions that should be taken into consideration while defining graphical representations that play an important role as a means of communication in the information systems development process. Various conventional models and methods are used to support the information systems development process, but still, after so many years of practice, information system projects continue to fail. Since the service-oriented modelling method provides a reasonable pedagogical support, it was analysed as one possible solution to the described design problems in the contexts of communication and leaning. The major contribution of this study is theoretical assumptions concerning communication, understanding and learning that influenced the development of the service-oriented modelling method. These assumptions were
taken into account in the process of defining the foundations of service-oriented modelling method.

### 6.3. Contribution of the Thesis

This thesis contributes with a new modelling method for information systems analysis and design across organisational and technical system boundaries, which is based on service orientation. *The Service-Oriented Modelling Method for Information Systems Analysis and Design* facilitates the semantic integration of the static and dynamic aspects of IS specifications. It provides graphical modelling techniques for bridging the pragmatic, semantic and syntactic specifications of service architectures. This method contributes to the field of IS conceptual modelling of information systems. The new service-oriented modelling method consists of three parts, which were introduced according to three research topics. These three parts are as following:

1. a pragmatic-driven specification of information systems,
2. a service-oriented modelling foundation for information systems analysis and design,
3. transition principles to implementation-specific design.

The first part of the method provides a pragmatic analysis, in terms of goals, problems and opportunities. It also suggests using a new way to refine pragmatic entities into service interactions. The second part of the method defines a new service-oriented modelling foundation. It provides a modelling notation and modelling process at a computation-neutral modelling level. The third part of the method introduces transition principles to implementation-specific design. Figure 87 presents the graphical overview of the contribution. The numbers define the publications, where the research results were presented, concerning every part of the method.
Figure 87: Overview of the contribution

All three main parts of the service-oriented modelling method contribute to IS analysis and design area in numerous ways, as listed below.

- A new way of pragmatic-driven specification of information systems. It provides pragmatic descriptions in terms of goals, problems and opportunities. Pragmatic specifications of service architectures are used as a driving force of system analysis and design. They aim to provide justification for conceptual representations of the enterprise’s organisational and technical components. The semantic specification of a problem is considered as a part of the actual situation. Semantic specification of a goal is considered as a part of the future situation. The modelling is regarded as a process of eliminating the problematic part of specification and complementing it by a desirable part of service architecture specifications. In pragmatic specifications, the opportunities are expressed by using the negative and positive influence dependencies with respect to designer’s problems and goals. The desired service architectures, which are characterised by goals, cannot be analysed without taking into consideration the related problems. Pragmatic specifications are gradually refined into service interaction loops between service requesters and service providers. The method supports a systematic way of modelling, starting from a business-oriented analysis ‘why’ to a service-oriented analysis. The creation of service-oriented diagrams is driven by pragmatic specifications.

- The service-oriented modelling foundation for information systems analysis and design includes a modelling notation and a modelling process. The modelling notation prescribes constructs that enables the integration of the static and
dynamic aspects of IS specifications. By integrating the structural, behavioural
and interactive aspects of conceptual models, the method helps to detect
inconsistencies, ambiguities and incompleteness among different modelling
dimensions. In other words, it helps to improve the quality of IS specifications
and therefore helps to reduce the communication gap among different
stakeholders involved. The service-oriented constructs are fundamental for
building the five main system analysis patterns (sequence, search, selection,
synchronisation and iteration) for service-oriented information systems analysis
and design. The modelling method is supported by the process, which helps to
combine the internal and external dimensions of service architectures. The
modelling process defines seven main seven steps of service-oriented modelling.

- The transition principles from computation-neutral to implementation-specific
design help to justify logical information system requirements. In such a way,
service-oriented conceptualisations can be used for the verification of software
architectures and for the semantic integrity control of object-oriented diagrams.

The presented service-oriented modelling method is an evolutionary method, which
provides a new way of enterprise modelling and integration. It is a method for
graphical design that enables reasoning about system architecture across
organisational and technical system boundaries. The service-oriented modelling
method is new in a variety of ways, listed below.

- The concept of service was explicitly used for conceptual modelling in
information systems analysis and design phase. It is expressed in terms of
traditional modelling elements such as actors, classes, attributes as well as
semantic relations such as interactions, transitions, inheritance and composition.

- The service-oriented foundation is built on the basis of the main constructs that
are rooted in the object-oriented methods. Nevertheless, the service-oriented
modelling process is based on a number of other principles such as
decomposition and separation of crosscutting concerns. The pragmatic
dependencies are used for the decomposition of service-oriented diagrams and
the principle of loose coupling between actors is used for the separation of
concerns. This principle is one of the key principles in managing the complexity
of IS specifications.

- The service-oriented way of modelling is computation-neutral. It provides the
transition principles from computation-neutral to implementation-specific
design. The transition principles were investigated for the object-oriented
models as well as for the Lyee methodology, which is used for computerised
software development.

- The service-oriented modelling method provides a better support for the
detection of discontinuities, incompleteness and inconsistency of IS
conceptualisations. Discontinuity can be identified by breakdowns in service
interactions. Incompleteness is detected by the help of underspecified goals,
which are not defined in terms of interactive, behavioural and structural aspects
of specifications. Inconsistency of specifications can be detected using a special set of inference rules (Gustas, 2010; Gustiené, 2003b).

- The method provides the possibility to combine the intersubjective and objective modelling traditions using a single diagram type, which helps to place focus on the integration of static and dynamic aspects. Various combinations of the static and dynamic dependencies are able to express the main workflow control patterns such as sequence, synchronisation, selection, search and iteration. Ignoring the static aspects in the pattern modelling research creates fundamental difficulties. If static aspects are not taken into account, then the quantity of patterns is too large and their usage becomes more complicated for business process change management. Five patterns are sufficient for service-oriented modelling, compared to the more than twenty patterns are used in BPMN (Russell et al., 2006).

The modelling notation used in the service-oriented modelling method is slightly different from the constructs used in the EM approach (Gustas & Gustiené, 2002; Gustiené, 2003a). Several modifications of enterprise modelling language (Gustas, 1997) were introduced in this thesis. They are the following:

- Only five static associations without semantic holes (only mandatory constraint) from at least one side of the association (Gustas & Gustiené, 2009b) are allowed. The definition of a semantic hole is the following: “if for any individual of concept B in the relationship with concept A does not exist an association to at least one instance of concept A then this is entitled as a semantic hole” (p.145) (Gustas, 1997). Semantic holes can cause incompleteness, inconsistency and ambiguity of system requirements, because if any concept in the relationship contains a semantic hole, then the relationship contains a hole as well. Therefore, only mandatory properties can be applied. The motivation behind that could be found in ontological foundations, stating that not having a property is not a property and that ‘null values’ have no meaning (Wand et al., 1999). Ambiguity of concepts can be identified and eliminated using completely applicable dependencies. These dependencies can guarantee semantic unambiguity.

- The aggregation dependency was excluded. Only the composition dependency is used in the service-oriented modelling method. Composition dependency, which is a stronger form of aggregation (Maciaszek, 2005) was defined more precisely and applied for service-oriented conceptual modelling. The composition has two additional constraints (Blaha & Rumbaugh, 2005): a constituent part can belong to one assembly and once a constituent part was assigned to the whole, it has the same lifetime as the whole. The composition dependency is useful for the formation of more abstract concepts that are viewed as parts and can be used for reasoning about system components; it defines how a whole relates to its parts. M:N (many-to-many) associations are not legal in the service-oriented modelling method. What this means is that a part cannot simultaneously belong to more than one whole. The aggregation dependency therefore could be seen as ambiguous and inference rules can not be applied. Composition dependency provides the possibility to propagate interactions from parts to compositional wholes.
• The interaction dependency is propagated upwards in the decomposition hierarchy. In service-oriented modelling the propagation of the interaction takes place from more abstract subsystems to more specific ones, which are viewed as organisational or technical system components. Inconsistencies between interaction links in different diagrams, and on different levels of abstraction, can be detected using inference rules of composition as well as inheritance dependencies. More specific diagrams are usually more detailed and less incomplete in comparison with high-level (more general) diagrams. As this way of modelling is incremental, it is possible that incompleteness of conceptualisations occur, but the most important thing is that there is a way, using inference rules of different dependencies, to identify deficiency, and go back to correct it. These modifications are very important characteristics of the service-oriented modelling language notation. It is one of the most important contributions to the development of a new service-oriented modelling method for analysis and design of information systems.

6.4. Justification of Service-Oriented Modelling Method

The method presented in this thesis is developed for the early and middle stages of the IS development life cycle that corresponds to analysis and very early design phases. In this thesis, the early stages of analysis are additionally supported by pragmatic models, which are traditionally understood as ‘why’ dimensions according to Zachman’s framework (1987). This dimension is analysed in terms of pragmatic entities. The service-oriented modelling method can be applied for the system analysis and design phases, which correspond to owner perspective and supports the ‘what’, ‘how’, ‘where’, ‘who’, and ‘when’ dimensions. The method also provides the transition principles to implementation-specific design, which corresponds to designer perspective in Zachman’s framework. The proposed service-oriented modelling method is an evolutionary approach for information systems analysis and design. It was not developed from scratch. The advantage of the method is that the modelling concepts used are familiar to system developers. It was developed taking into account the traditional approaches to information systems modelling such as object-oriented approach with UML, which is widely used for software modelling. The object-oriented design events are fully expressed in terms of creation, termination and reclassification actions of the service-oriented modelling method. Pre-condition and post-condition classes that represent the objective perspective in the service-oriented modelling method are similar to the entities of the Entity Relationship data model (Chen, 1976) and represent data at rest. In the Structured System Analysis and Design approach (DeMarco, 1979; Gane & Sarson, 1979) data at rest is represented by data stores, which may take the form of different physical representations. Many process modelling similarities can be found with data flow
diagram, which represent moving data between external entities, processes and data stores within IS. As interaction and structural characteristics are of no interest for data flow diagram (Hoffer et al., 2004), it is difficult to track continuity of the business process. A data flow diagram does not support the five analysis patterns, which are introduced in this thesis and the service-oriented modelling method can support. Any concept of the service-oriented diagram can be viewed as a class defined in terms of attribute dependencies, which are important for defining semantic differences among concepts. In service-oriented diagrams, the distinction is made between three kinds of flows: information, physical and decision flows, which are not defined in DFDs. The service-oriented way of modelling is pragmatic-driven and computation-neutral. The motivation is listed below.

- Pragmatic aspects drive the overall service-oriented analysis and design process. Comprehensive business system engineering can not separate the pragmatic details of service usage from the semantic description of service architecture. A holistic understanding of the semantic and pragmatic dependencies between service requesters and service providers includes an analysis of service objectives and a conceptualisation of their interaction links. Pragmatic dependencies, used for the analysis of pragmatic aspects of services, help to identify possible contradictions between desirable and undesirable situations. Goal decomposition and refinement links help system designers to understand crosscutting concerns of enterprise architecture. Pragmatic hierarchies are necessary for the detection and elimination of the problematic fragments of specifications and for bridging to the desirable fragments. A systematic modelling way of IS analysis from a pragmatic to a semantic level provides the desired motivation and traceability of requirements. It allows for a tracing requirement changes, which are essential for the maintenance of a high quality of software, and for systematic change management.

- The modelling process is computation-neutral, because pragmatic and semantic levels are not influenced by any implementation details. The semantics of computation-neutral models are expressed by extended graphical specifications of interactions between service requesters and service providers. Such conceptual models are more comprehensible for business experts with no technical background.

The service-oriented modelling method supports the architectural principles of system design, where the system’s architecture is the fundamental organisation of a system embodied in its components. Enterprise architecture is defined using a three-level framework, modelling process and techniques necessary to understand the system as a whole. Additionally, it provides the principles of transition to implementation-specific representations.

The uniqueness of the service-oriented modelling method is that it is based on the principles of service orientation. Business processes can be analysed as a composition of service interactions. Any business process fragment is viewed as a part of service architecture. Service-oriented representations are built by conceptualising interactions among organisational and technical components, which are viewed as various types of enterprise actors. By matching the actor dependencies from service requesters to
service providers, one can explore possibilities, rights and responsibilities that are predefined to different types of actors. Continuity of interaction loops is the main principle of service orientation.

A service-oriented paradigm provides the possibility to integrate intersubjective and objective system analysis traditions. The intersubjective perspective is based on interaction relationships among actors. Every interaction is defined in terms of noteworthy changes of the objective perspective. The objective perspective defines object transitions, when the actions take place. The cohesion of intersubjective and objective perspectives results into a single modelling notation, which allows for the integration of static and dynamic aspects. The presented set of semantic dependencies was sufficient for defining the five main service-oriented analysis patterns, namely sequence, synchronisation, search, selection and iteration.

The integration of internal and external behaviour, which is encapsulated in the service concept, creates challenges for the traditional object-oriented methods for analysis and design, as well as for business process modelling languages such as Business Process Modelling Notation (White, 2004). Various types of object-oriented models are used for the semantic modelling of business processes, but every model typically focuses on a single dimension. Since different dimensions are intertwined, it is important to maintain integrity across multiple graphical representations.

One of the biggest advantages of service-oriented modelling for information systems analysis and design is that an enterprise system can be viewed and analysed as a collection of subsystems. Every subsystem can play roles of service requesters and service providers, which can be specified according the following principles:

- subsystems are loosely-coupled. The dynamic aspect of subsystems is expressed using interaction dependency, which is gradually refined in terms of structural and behavioural dependencies.
- subsystems are non-overlapping in functionality. They are autonomous components of an enterprise system.
- subsystems are unambiguously decomposed. Only instantiation, composition and inheritance dependencies are used to define components on the lower level of abstraction. The aggregation relation is ambiguous and therefore is not used in service-oriented modelling method.

Using these main service-oriented principles, make enterprise systems easier to comprehend, analyse, design and maintain. A new service-oriented foundation for information systems analysis and design is centred on modelling the semantic and pragmatic aspects of business processes. The computation-neutral modelling of business processes, in terms of service architectures, provides the possibility to view IS analysis and design as an engineering discipline, which provides a basis for semantic integrity, continuity and completeness control.
6.5. Implications for Different Target Groups

To understand the value of the proposed service-oriented modelling method, it is necessary to look at it from the perspectives of different stakeholders. The target groups or ‘specialists’ of the new method proposed in the thesis are outlined below.

- **Enterprise architect** - responsible for the alignment of business and IT operations. Enterprise architect is also responsible for overall integrity, continuity of service architectures as well for coherence between business data and business processes. The presented systematic modelling method, together with the modelling techniques and rules, can be used to comprehend and build a holistic view of strategic and operational assets of an organisation. Being computation-neutral, service-oriented representations can be used as a communication tool among all stakeholders involved.

- **Business analyst** - responsible for the overall business analysis process with the goal to identify business needs and determine solutions to business problems. The service-oriented modelling method contributes to the following tiers of business analysis: the application of pragmatic-driven approach is relevant to strategic planning; to the conceptualisation of structural and behavioural aspects of service architectures; and the verification of technical solutions according to strategic needs stated at a pragmatic level.

- **Project manager** - must understand and lead the entire system development process. The service-oriented modelling method is useful as it provides an incremental and iterative development process from early requirements analysis to implementation-specific design. The three-level modelling framework and modelling techniques provides a way to understand service architectures on different levels of abstraction and to comprehend enterprise system architecture as a whole.

- **System designer** - one of the main target groups of this method. The method provides a constructive and systematic way to justify and validate solutions by business goals. It helps to control incompleteness of design by tracing original requirements. The method provides the modelling language, process and techniques, which support the analysis and design of service architectures. The method enables the analysis of services on different levels, breaking them into smaller parts to address the autonomy and crosscutting concerns of requirements.

- **Academic and industrial researchers** - involved in developing new methods and theories. This method contributes with entirely new principles of IS analysis and design. It helps to develop new modelling ways for solving different semantic integration problems in conceptual modelling of IS.
6.6. Conclusions

The ultimate goal of this thesis is the Development of a New Service-Oriented Modelling Method for Information Systems Analysis and Design. This method is new, because it is based on modelling service interactions, which allow the integration of structural and behavioural aspects of IS conceptualisations. The method includes the modelling process, which is based on service-oriented principles. The developed modelling process is centred on the construct of service. Service orientation is important for the separation of crosscutting concerns and for an analysis of business process continuity. Therefore, the new modelling method facilitates reasoning about semantic integrity and helps to manage the complexity of IS specifications. The purpose of this method is to provide a modelling process and modelling techniques for early requirements analysis, where the pragmatic and semantic aspects of service architectures can be analysed together. The modelling process supports the traceability of pragmatic specifications into conceptual representations of service architectures. The achievement of this goal was reached through three sub-goals, which correspond to three research topics. The topics are as following:

• pragmatic-driven specification of information systems,
• service-oriented modelling foundation for information systems analysis and design,
• transition principles to implementation-specific design.

The results of each research topic gave the answers to stated research questions of the thesis and contributed to the development of a new service-oriented modelling method. The research studies concerning the first research topic answer the following research question of the thesis: How can a conceptual modelling process be driven by pragmatic considerations? Answering this question resulted in the development of a pragmatic-driven method of information systems analysis and design, including the principles and techniques of pragmatic analysis as well as a notation and modelling language. The advantage of the pragmatic-driven approach for the service-oriented modelling method is that any business process fragment can be defined in terms of service architectures. A service specification from the pragmatic point of view is analysed as a problem, goal or opportunity. Pragmatic descriptions aim to provide justification for change management of enterprise components, which are defined in terms of interactions between service requesters and service providers. Strategic analysis and business solutions can be represented by a set of pragmatic dependencies, which drive and motivate the overall IS design process. A pragmatic-driven approach supports the principle of separation of concerns, which is one of the main principles for managing the complexity of enterprise system specifications. Goal decomposition hierarchy is important as it describes how various high-level objectives are going to be achieved. Goal analysis helps to identify new business processes and provide a basis for reasoning about the semantic completeness of system specifications. Goal hierarchies are used to specify the desirable changes that are represented at the
semantic level in terms of service interactions between different organisational and technical components. The pragmatic-driven approach enables the mapping of pragmatic modelling dimensions to conceptual representations, which define the semantics of business processes.

The second research topic answered the following research questions of the thesis: How can the concept of service be used explicitly for the analysis and design of information systems? How can the static and dynamic aspects of information systems specifications be integrated at the conceptual level? The answer was given by presenting the service-oriented modelling foundation. The basis for this foundation includes the method, principles and techniques of service-oriented analysis; service-oriented modelling process notation as well as modelling language. Modelling constructs were used for the development of five analysis patterns. It was demonstrated that various combinations of static and dynamic dependencies using service-oriented modelling constructs are able to express the main workflow control patterns such as sequence, synchronisation, iteration, selection and search. Traditional pattern research uses only dynamic models, which are expressed using UML activity, state diagrams, Petri-nets (Russell et al., 2006) or business process diagrams (BPMN, 2004; White, 2004). These modelling techniques do not take into account the static aspects of business processes. If the static aspects are not taken into consideration, then analysis patterns become much more complex. This situation causes difficulties in business process change management and evolution.

A new service-oriented modelling method is based on the merging of intersubjective and objective traditions of information systems analysis and design. It enables the integration of the static and dynamic aspects, which is important for the control of semantic integrity. It integrates various dimensions of service-oriented architecture into a single meta-model. The advantage of service-oriented principles is that they facilitate semantic integrity control between the static and dynamic aspects, which are important to maintain a holistic representation, where external and internal views of service conceptualisations are visualised together. The service-oriented foundation supports reasoning and understanding of IS specifications and technical system boundaries.

The third research topic answered the following research question: How can service-oriented conceptual representations be aligned with implementation-specific design? Service-oriented modelling method is computation-neutral. Diagrams follow the basic conceptualisation principle in representing only computation-neutral aspects that are not influenced by any details of implementation solutions. Computation-neutral representations help to justify logical software representations that are expressed at the syntactic level. Conceptualisations of service architectures were used as a tool for the verification of implementation-specific design. It was demonstrated how the service-oriented method can be applied for development of object-oriented diagrams.

Semantics of state changes were compared with creation, termination and reclassification actions of service-oriented diagrams. It was demonstrated that service-oriented diagrams are capable of capturing behavioural events and effects, which are represented in various UML diagrams. The effects of interactions are
difficult to trace using object-oriented approach. Service-oriented diagrams are able not only to describe actions, but also to capture control conditions on classes of objects, which are important for triggering other actions. It was also shown how conceptual representations can be implemented using a set of loosely-coupled organisational and technical components. Organisational system in the presented case studies is supported by a technical system part, which is conceptualised in terms of software components. UML is widely used as a standard in many companies, it is important to show how UML diagrams can be motivated using a computation-neutral way of service-oriented modelling. Service-oriented diagrams were used for understanding and explaining of object-oriented diagrams, which are implementation-specific representations. The semantics of service-oriented constructs are more concise as they deal only with computation-neutral aspects, which have no implementation bias. Being computation-neutral, conceptual representations of service architectures are less complex. Since computation-neutral representations are easier to comprehend for business experts as well as system designers, they facilitate understanding and learning. Thus the developed service-oriented method can be used for bridging the communication gap among various types of stakeholders involved in IS analysis and design process.

The changes of service architecture need to be constantly visualised and agreed upon. It is critical to align business design with IT system architecture design in order to make both organisational and technical system parts work more effective. The presented foundation for defining conceptual representations of service architectures can be used for a systematic analysis of the enterprise system changes. The service-oriented modelling method helps to bridge various dimensions of enterprise system architecture. This method could be seen as a tool in diagnosing potential problems and managing changes in organisations. Service architecture specification is a basis for reasoning about semantic incompleteness, inconsistency, discontinuity and ambiguity of IS specifications. The use of a single model, as proposed in the method, makes it easier to validate and verify graphical representations on different levels of granularity. It helps to detect discontinuity of IS specifications.

The proposed modelling method is based on new principles of service-oriented analysis and design. It provides a possibility to analyse enterprise architecture as a composition of services. A new service-oriented modelling method facilitates reasoning about the integrity of IS specifications across organisational and technical system boundaries. It contributes to the principles of incremental design, which inspire various IS design decisions. The method provides the basis for gradual and systematic understanding of how pragmatic, semantic and logical IS requirements are linked together. The possibility to detect and eliminate undesirable characteristics of IS conceptualisations helps to improve communication among stakeholders.

The foundation of the presented service-oriented approach is based on theoretical assumptions concerning language, communication and understanding. These assumptions were taken into account while constructing a modelling language for service-oriented modelling method. The modelling language is based on service-oriented principles, which are used in both the analysis and design phases. It has the
potential to reduce system architecture evolution complexity and to improve learning and comprehension. The service-oriented modelling method supports early requirements analysis, in terms of goals and problems, which allows for tracking of requirement changes during the development process. A service-oriented paradigm should bring significant benefits, including improved ability for organisations to maintain strategic knowledge in a systematic way, reduced costs for a systematic analysis of new IT solutions, before they are implemented, improved integrity and traceability within companies by providing comprehensive service architectural descriptions. The presented graphical service-oriented modelling method can be seen as a new way of modelling computerised information systems that span across organisational and technical system boundaries.

6.7. Future Research

The presented service-oriented modelling method has several unique advantages such as unambiguous principles for the separation of crosscutting concerns and semantic integration principles. Nevertheless, these advantages cannot be attributed in a straightforward way to UML or to any other industrial IS development standard. The problem is a paradigmatic mismatch between the service-oriented modelling foundation and object-oriented methods. The presented case studies included some tests of transition principles from computation-neutral service-oriented diagrams to implementation-specific design. Nevertheless, the mapping rules to UML and other standards need to be defined more precisely. UML individual diagrams are clearly defined, but an integrated semantics among models is still missing. The static and dynamic aspects of IS specifications are complementary and they cannot be analysed in isolation. The integrity control rules should be possible to introduce directly into the UML meta-model. Particular views and specific types of diagrams, which define structural, behavioural or interactive aspects, should be possible to generate by producing projections of an integrated model. Currently:

• decomposition principles only help to validate whether system specifications fulfil their intended purpose in relation to service-oriented diagrams, which are not directly linked to object-oriented design,
• the method helps to integrate structural, behavioural and interactive aspects of IS conceptual representations by using a special diagram type, which is based on slightly different constructs than UML notation.

Despite the presented advantages, the newly developed method cannot solve, in full scale, all semantic integrity problems of conventional approaches. Still, there is a lot of work to be done. Even if the presented advantages of the service-oriented modelling method are quite significant, they still cannot yet be viewed as a complete solution to the problems listed below.
• **Interaction flows cannot be explicitly captured in the traditional approaches.** Enterprise systems span across organisational and technical system boundaries. It is difficult to keep track and of crosscutting concerns by using UML diagrams. The service-oriented method helps to separate crosscutting concerns and to support continuity control of business process representations, which are based on modelling interaction flows. Typically, system designers are not used to conceptualise information systems in such way. Such a way of modelling requires strict discipline and good modelling skills.

• **Integrity problem of UML diagrams:** A holistic understanding of service architectures requires the integration of different IS architectural dimensions. Since various modelling dimensions are highly intertwined, it is crucial to maintain integrity across multiple diagrams. If static and dynamic aspects are analysed in isolation, it is difficult to detect integrity problems of information system specifications. The service-oriented modelling construct helps to integrate various dimensions of IS specifications at the semantic level, but integration principles of UML diagrams at implementation-specific level should be still developed.

• **Formalisation problem:** IS architecture can be perceived from different viewpoints by various stakeholders and can be represented on various levels of abstraction. A consistent way of dealing with strategic, business and implementation-specific conceptualisations is missing in most traditional modelling approaches. The decomposition principles of the service-oriented modelling method help to detect inconsistency of IS specifications. Nevertheless, the mapping rules to object-oriented diagrams still need to be formalised.

• **Change management problem:** There are no stopping rules defined in the proposed service-oriented modelling process. On the other hand, such rules are missing in the conventional methodologies of information systems analysis and design as well. Pragmatic specifications can be used by CASE tools to control over-specification or under-specification of graphical descriptions, which represent business processes and business data. Such tools should be extremely useful for business managers with no technical design background. Changes in service architectures should be managed in a systematic way. The integration and decomposition principles of the pragmatic-driven service-oriented modelling method should serve as a firm theoretical basis for a change management of information system conceptualisations.

In this thesis, only functional requirements of IS are taken into account. Future research should therefore be devoted to how non-functional requirements can be integrated in this method. Automatic control of IS specification quality such as inconsistency, incompleteness and ambiguity, require the development of a special set of CASE tools. The service-oriented modelling foundation and transition principles to implementation-specific design, which have been introduced in this thesis, are not sufficient. Specific mapping rules to UML and other standards still need to be defined more precisely. An important area of future work also includes the development of the automated guidance from the pragmatic to the semantic levels of IS.
specifications. Goal-oriented analysis should deal with the modelling of individual actor goals. Currently, only system design goals are covered in this thesis. The presented service-oriented method is focused only on conceptual dependencies between actors, but not on their goals. This type of future work is more relevant for the area of artificial intelligence, as actors can be seen as technical agents and their goals are usually prescribed by other organisational components.