8 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

A Service Description Method for Service Ecosystems

Meta Models, Modeling Notations, and Model Transformations

von Gregor Scheithauer

UNIVERSITY OF BAMBERG PRESS

Schriften aus der Fakultät
Wirtschaftsinformatik und Angewandte Informatik
der Otto-Friedrich-Universität Bamberg

**Schriften aus der Fakultät
Wirtschaftsinformatik und Angewandte Informatik
der Otto-Friedrich-Universität Bamberg**

Band 8

University of Bamberg Press 2011

A Service Description Method for Service Ecosystems

Meta Models, Modeling Notations, and Model Transformations

von Gregor Scheithauer

University of Bamberg Press 2011

Bibliographische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Informationen sind im Internet über http://dnb.ddb.de/ abrufbar

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität als Dissertation vorgelegen
1. Gutachter: Prof. Dr. Guido Wirtz
2. Gutachter: Prof. Dr. Tim Weitzel
Tag der mündlichen Prüfung: 25.02.2011

Herstellung und Druck: Digital Print Group, Nürnberg
Umschlaggestaltung: Dezernat Kommunikation und Alumni

ISSN: 1867-7401

ISBN: 978-3-86309-007-4 (Druckausgabe)

eISBN: 978-3-86309-008-1 (Online-Ausgabe)

URN: urn:nbn:de:bvb:473-opus-3151

Acknowledgements

This work would have been impossible without the support and guidance of numerous people. I am especially thankful for Professor Dr. Guido Wirtz for supervising this thesis and for his support, guidance, and constructive ideas over the course of my research. Equally, I would like to thank Professor Dr. Tim Weitzel and Professor Dr. Christoph Schlieder for their support as members of my dissertation committee.

I show my gratitude to the Knowledge Management team at Siemens Corporate Technology, and especially to Hermann Friedrich who provided the environment and opportunity to follow my research interests. Likewise, I would like to thank my master student Hao Hu for all his work that supported my research project. Additionally, I also would like to thank my colleagues from the Theseus/TEXO research project for years of great collaboration on service ecosystems and service engineering. Furthermore, I am grateful for my friends Philipp Zeidler and Jan Gross for proof reading my thesis.

Above all, I am indebted to my parents, Dagmar and Manfred Scheithauer, to my brother Robert Scheithauer, and to Helen Steinfelder. This work would have been impossible without their constant encouragement.

Gregor Scheithauer

For Helen

Contents

List of Tables

List of Figures

Part I.

Foundations

1. Introduction

Globalization, technological change, and an increasing demand for services [106] transform countries from industrial economies toward service economies. Regarding this trend, it becomes clear that services and their development play an important role in today's and tomorrow's business endeavor. In line with this trend, service ecosystems have emerged as an evolution of service orientation [105] that takes services from merely addressing integration purposes to the next level by making them available as tradable products on service market places [18]. Examples for existing market places include *StrikeIron* and *SalesForce.com*. Services and their description in terms of definitions, marketing, management, documentation, communication, implementation, simulation, and verification is an overarching discipline that spans business and information technology (IT).

The business perspective concentrates on definitions, marketing, and management of business services [22, 45, 66, 71, 73, 137]. In general, business sciences define services as a process with an intangible outcome that does not result in ownership [66]. Kotler and Keller [66], for example, define services as "*... any act or performance that one party can offer to another that is essentially intangible and does not result in ownership or anything. Its production may or may not be tied to a physical product*". Whereas service marketing includes pricing strategies, promotion, and positioning [24], service management fosters the integration of employee roles and systems delivery, and the balancing of demand and capacity [45, 66, 71].

The IT perspective, on the other hand, investigates implementation, protocols, simulation and verification of distributed computational functionality in terms of web service technology [10,26,96,138]. From this perspective, a service is described as a software system that enables machine communication using network protocols [138]. Recently, the design of distributed software systems has followed the principles of Service-oriented Architecture (SoA) [105], which implies that services are (1) technology neutral, (2) loosely coupled, and (3) transparent regarding their location. This is achieved with a set of standards in web service technology, such as transport protocols (e.g. HTTP), messaging protocols (e.g. SOAP), description protocols (e.g. WSDL), and discovery protocols (UDDI) [138].

Business modeling, in general, aims at aligning business and IT and in particular conceptual modeling and generating IT artifacts [33]. According to Winter et al. [143], conceptual modeling comprises the development of models and methods in order to express business requirements. These models support formal documentation, universal communicating, and analyzing business requirements. Indulska et al. [53] assert the following benefits for the special case of modeling business processes: a greater ability for process improvement, consistent understanding of processes, and improved communication of business processes. Generating IT artifacts from conceptual models implies the existence of a set of formal rules that can be applied for the generation process. These generation processes run either automatically or semi-automatically, which results in a faster development process, fewer errors, and easier change management [124]. According to Indulska et al. [53], model-driven process execution is one of the top-five benefits of process modeling.

From the viewpoint of information systems, various methods and models for documentation, communication, and analyzing already exist. Gordijn [43] developed the e^3 Value ontology in order to document and analyze e-commerce business models. Business Process Modeling Notation (BPMN) [93], on the other hand, is a modeling notation that allows documenting and analyzing business processes. Furthermore, a wealth of approaches exists to generate IT artifacts from BPMN [102,109,135], namely

Business Process Execution Languages (BPEL) [10].

While there is work available on modeling business processes and aligning them with information technology, this work aims at closing the gap for modeling business services and aligns them with the numerous technical standards. This thesis addresses the development of models and methods for documenting, communication and analyzing service descriptions, and the generation of standard Web service artifacts.

The following sections discuss the motivation for this research project, its research questions, its general contribution, classify this approach into Service-oriented Architecture, the structure of this thesis, and a list of available publications.

1.1. Motivation

The gist of service ecosystems is to enable trading of services over the Internet between different legal bodies, business services composition, and building platforms for IT-supported service provisioning [56]. This development raises the need for rich service descriptions. Figure 1.1 depicts service trade in service ecosystems where service descriptions have a significant part (cf. [67]). By means of service proposition, service providers advertise their services toward potential consumers, whereas during discovery and selection, service consumers specify their service preferences to the provider. If a service consumer selects an appropriate service, providers and consumers negotiate and finally agree on service levels (SLA), which are monitored throughout value exchange. In the event of service levels not being met, compensation has to be triggered. During service profiling, valuable information on services' performance is stored, which is gathered through value exchange and monitoring.

In order to enable service trade, a shared and common understanding of services must become available. From the perspective of service providers, a business-orientation of service descriptions becomes a cru-

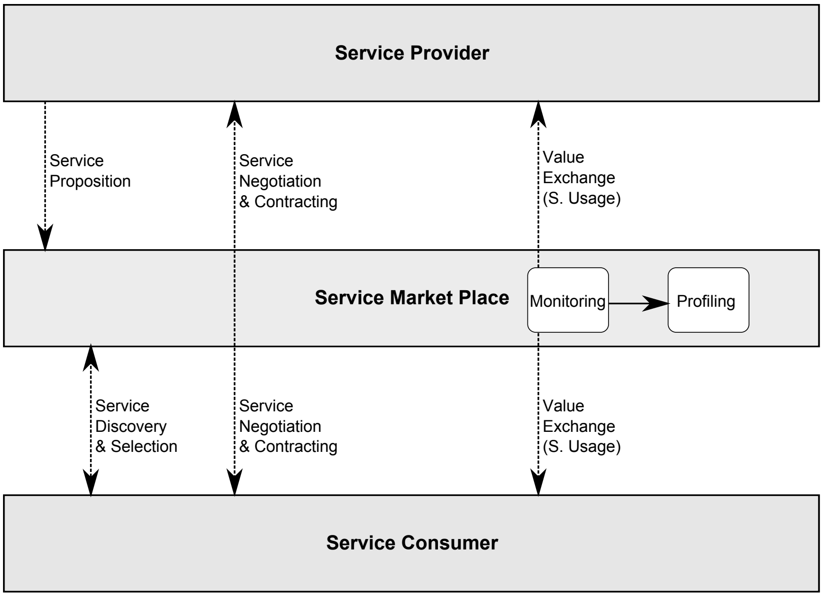

Service Provider
Service Proposition
Service Negotiation & Contracting
Value Exchange (S. Usage)
Service Market Place
Monitoring
Profiling
Service Discovery & Selection
Service Negotiation & Contracting
Value Exchange (S. Usage)
Service Consumer

Figure 1.1.: Trade in Service Ecosystems

cial part in the service development process, which is impeded for the following reasons. Firstly, there does not exist a formalism for defining service descriptions on a conceptual level [35, 67, 126]. Secondly, service descriptions embody divergent information and need the involvement of different subject-matter-experts for creating and understanding service descriptions. Thirdly, ample *overlapping* technical specifications exist that describe web services with first-order logic, predicates, and XML, such as Web Service Description Language (WSDL) [27], Web Service Modeling Ontology (WSMO) [112], and Semantic Annotations for WSDL and XML Schema (SA-WSDL) [37], which are not easily derived from conceptual models and maintained. Fourthly, there is no real alignment between service business models and IT-related service descriptions. These reasons indicate that the service description development process is prone to errors, slow, and irreproducible. While recent work has concentrated on *business process modeling* with a focus on how to formalize the relationship between conceptual business requirements and how to implement them with the help of service-oriented architectures (cf. [102]), no attempts have been made for enhancing the process of providing service descriptions.

Service providers would benefit from a method that would allow documentation, communication, and reasoning of service descriptions on different levels of abstraction, and which would provide a much more efficient development process.

1.2. Research Questions

The basic question that this work attempts to address is how to describe services in a business-oriented fashion in order to leverage service proposition and service discovery. It can be subdivided into the following questions.

[RQ1] Which service properties are relevant for service ecosystems?

As aforementioned, service descriptions lack a formalism [35, 67, 126] for service proposition as well as service discovery. Naturally, such a formalism can be seen from a technical as well as from a business perspective. An appropriate common set of functional and non-functional properties that is valid for both perspectives might encompass enough richness for service matchmaking and embodies a technical representation to enable service trade over the Internet.

[RQ2] How are service properties to be modeled?

Business modeling is a discipline that uses graphical languages, e.g. UML and BPMN, to elicit, to document, to communicate, and to reason about business requirements [53]. That is particularly important for business modeling as it usually involves many experts with different backgrounds and expertise. A dedicated modeling notation for service properties would enable the application of a common set of properties.

[RQ3] How are standard web service artifacts generated from service properties?

While there exist numerous technical languages to implement web service descriptions, their development is not aligned with business requirements, slow and incomprehensible [124]. While the common set of service properties and the dedicated modeling notation are a possible abstraction of these technical languages, the development of guidelines, techniques, and transformation rules would overcome these additional issues.

1.3. Own Contribution

This section introduces an approach to address the challenges mentioned in section 1.1 and how to answer the questions raised in section 1.2. The final outcome of this thesis is called *Service Description Method for Service Ecosystems (SDM4SE)*. It allows describing electronically consumed services, offers modeling facilities, links professional and technical theories, and provides a method that supports domain experts. The aim of this thesis is to explore descriptions for services which can be traded over the Internet, service descriptions' conceptualizations, its technical representations, and available methods. In the course of this thesis, current shortcomings will be identified and discussed. This research approach follows the information system research cycle of Hevner and March [49]. Design science in general follows a five-step process: (1) Problem awareness, (2) suggestion, (3) development, (4) evaluation, and (5) conclusion. The *Foundation* part of this thesis addresses the problem awareness. Whereas the *Design* part suggests and develops a solution, the *Implementation and Evaluation* part offers an evaluation of the suggested solution. Finally, part *Finale* offers a conclusion. The course of action is subdivided into three steps, which the following paragraphs describe briefly.

Service Properties A reference model is developed with different abstraction layers for service description modeling. Following that, a set of requirements for services descriptions is derived from the service ecosystem domain as well as from a literature research. Then, meta models for each reference model layer to formalize a valid service description are developed. For doing so, existing literature is investigated about functional and non-functional service properties. Sources are versatile and include IT standards (ebXML [80], Dublin Core Meta Data [6], IEEE 830-1998 [127]) and academic publications (O'Sullivan [99], Barbacci et al. [17]).

Modeling Languages In order to apply and to use the identified service properties, UML Profiles [87] are developed for each service property meta model [40]. For doing so, existing modeling languages are investigated, including the e^3 Value ontology [107], UML Profile and Metamodel for Services (UPMS) [85], and UML Profile for Modeling Quality of Service (UPMQoS) [92]. Furthermore, a guideline is developed in order to support subject-matter-experts to apply the modeling notations by examining work in the area of service marketing [22, 64, 66, 73].

Standard Web Service Artifact Generation For generating standard web service artifacts, the identified service properties are mapped to existing web service standards. OMG's Model-driven Architecture (MDA) [81] approach is employed along with model transformations [89]. While possible standards include WSDL [27], Universal Description Discovery and Integration (UDDI) [96], WSMO [112], Web Ontology for Services (OWL-S) [74], and Web Service Level Agreements (WSLA) [59], this thesis concentrates on WSDL to show the feasibility of artifact generation.

1.4. Classification of this Work

This section explains how this work is embedded in the general field of Service-oriented Architectures (SoA). This integration supports the reader for a better understanding of the addressed problems, the offered solutions and its significance in the SoA domain. Figure 1.2 displays an adapted version of Papazoglou's Extended SoA Model that is used for the integration.

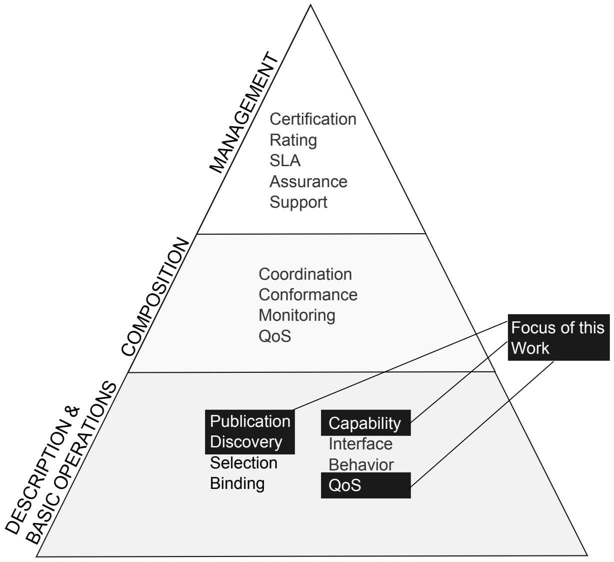


Figure 1.2.: Extended Service-oriented Architecture (cf. [105])

The basic SoA Model comprises three stakeholders: (1) Provider, (2) Client, and (3) Registry, as well as the three functions: (1) Bind, (2) Find, and (3) Publish. Papazoglou [105] realized that this basic model does not reflect the complexity of SoA. In consequence, he developed an extended model for SoA that is illustrated in figure 1.2. This model serves as a basic understanding for challenges in the SoA domain and acknowledges challenges in management, service composition, coordination, and security.

However, it does not provide answers for these challenges. Papazoglou divides the model into three layers: (1) a service management layer, (2) a service composition layer, and (3) a basic layer addressing service descriptions and basic operations. Whereas the *service management layer* relates to certifications, ratings, service level agreements, assurance, and support, the *service composition layer* focuses on coordination, conformance, monitoring, and quality of service.

This work contributes to the *basic layer*, and in particularly to the areas of capability, quality of service, publication, and discovery. In general, capabilities and quality of service can be interpreted as functional and non-functional properties. The aforementioned service properties will allow expressing services' *capabilities* and *quality of service*. Moreover, the modeling languages enable service providers to document service offerings for communication and reasoning, which support the service *publication process*. The generation of standard web service artifacts, such as WSDL and UDDI, enables the *discovery process*.

1.5. Context of this Work

This work was conducted while I worked for Siemens Corporate Technology, Siemens's research and development unit. In particularly, I was working for the Application Integration competence center in the Knowledge Management department, where I was involved in the following topics: Business Process Management, Service-oriented Architectures, Semantic Web Services, and Model-driven Design. Some results were developed while I managed and contributed to two research projects: (1) ProCHeSO and (2) Theseus / TEXO [5]. The Process Composition of Heterogeneous Service Orchestrations (ProCHeSO) lasted two years. It aimed at integrating web services with a different granularity: technical services and business services. The attempt was to use Semantic Web Services to offer a common language to describe both services types. Especially, the

Web Service Modeling Ontology [112] was applied. Theseus / TEXO is a government-funded research project, spanning four years, with more than 13 organizations and more than 50 researchers involved. Theseus / TEXO aims at realizing the Internet of Services (IoS). The research agenda spans business models, communities, user experience, governance, service innovation, service engineering, service usage, and service delivery. My work focuses on model-driven service engineering and service descriptions.

1.6. Outline

This work is structured into four parts as shown in figure 1.3.

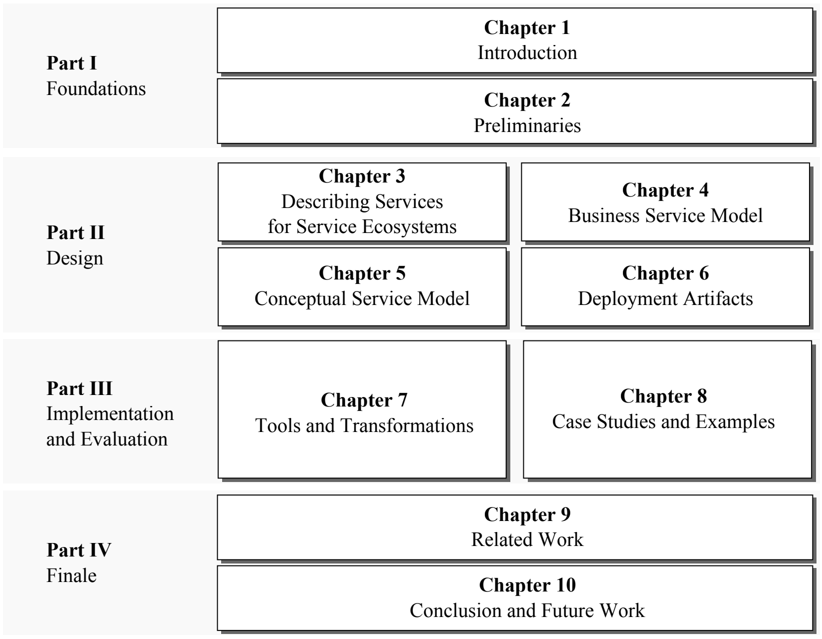


Figure 1.3.: Structure of this Work

PART I While chapter 1 introduced the motivation for this research along with its research question and approach, chapter 2 presents an overview of preliminary work which is the basis for this research: Basic concepts and technologies, service classifications, service ecosystems, and existing approaches for describing services.

PART II Following the foundations, chapter 3 presents the Service Description Method for Service Ecosystems (SDM4SE). At first, it offers further insights about requirements for a service description as well as about the approach taken in this work. Subsequently, the chapter presents the service description reference model on top of the Open-EDI reference model and method engineering. This reference model is the basis for developing method artifacts for a business layer, a conceptual layer, and for creating deployment artifacts. Finally, it introduces a motivating example that is used in the subsequent chapters. Chapters 4 and 5 present the method's first and the second layer in detail. They contain research about business models and service marketing as the basis for the two meta models. Also, these chapters show the development of two novel modeling notations. Chapter 6 presents available specifications for describing services in a technical manner. Furthermore, it shows the development of one abstract mapping for automatic generation of WSDL.

PART III After the design of SDM4SE, chapter 7 presents two tools which support the service description development process. Finally, the chapter describes the technical setting and the implementation of model-to-model transformations. For the evaluation of SDM4SE and its integration into ISE, chapter 8 presents two case studies. One case study was conducted in the IT outsourcing domain and the second case study was carried out in the insurance domain.

PART IV Chapter 9 discusses work that is related to describing services and to service engineering. Finally, chapter 10 provides a summary of this work, presents answers for the research questions, and offers insights about future work.

1.7. Publications and Research Development

Parts of this thesis have been published at international academic conferences, workshops, and doctoral consortiums. This section lists these publications during the three years of the thesis. The publications include service description meta models, novel modeling notations, and case studies.

2008

- SCHEITHAUER, G. [114] Process-oriented Requirement Modeling for the Internet of Services. In *Proceedings of the 1st Internet of Services Doctoral Symposium 2008 (I-ESA)* (Berlin, Germany, March 25, 2008), R. Ruggaber, Ed., vol. Vol-374.

- SCHEITHAUER, G., AND WIRTZ, G. [123] Applying Business Process Management Systems – a Case Study. In *The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE'08)* (Redwood City, California, USA, July, 1 - 3 2008), pp. 12–15.

- SCHEITHAUER, G., WIRTZ, G., AND TOKLU, C. [125] Bridging the Semantic Gap between Process Documentation and Process Execution. In *The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE'08)* (Redwood City, California, USA, July, 1 - 3 2008).

- KETT, H., VOIGT, K., SCHEITHAUER, G., AND CARDOSO, J. [62] Service Engineering in Business Ecosystems. In *Proceedings of the*

XVIII. International RESER Conference (Stuttgart, Germany, September, 25 - 26 2008).

- Scheithauer, G., and Winkler, M. [122] A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik 78, Bamberg University, October 2008. ISSN 0937-3349.

- Winkler, M., Cardoso, J., and Scheithauer, G. [142] Challenges of Business Service Monitoring in the Internet of Services. In *iiWAS'2008 - The Tenth International Conference on Information Integration and Web-based Applications Services* (Linz, Austria, November, 24 - 26 2008), books@ocg.at, Austrian Computer Society, pp. 613–616.

- Scheithauer, G., Augustin, S., and Wirtz, G. [116] Describing Services for Service Ecosystems. In *ICSOC Workshops* (Sydney, Australia, December 1, 2008), G. Feuerlicht and W. Lamersdorf, Eds., vol. 5472 of *Lecture Notes in Computer Science*, Springer, pp. 242–255.

2009

- Scheithauer, G., Augustin, S., and Wirtz, G. [118] Service Value Properties for Service Ecosystems: A Reference Model and a Modeling Guideline. In *Proceedings of the EOMAS Workshop* (Amsterdam, Netherlands, June 8-9, 2009), J. Barjis, J. Kinghorn, S. Ramaswamy, E. Dubois, and P. Johannesson, Eds., vol. Vol-458 of *CEUR-WS*.

- Scheithauer, G. [115] Business Service Description Methodology for Service Ecosystems. In *Proceedings of the CAiSE-DC'09 16th Doctoral Consortium held in conjunction with CAiSE'09 Conference Ams-*

terdam, The Netherlands, June 9-10, 2009 (Amsterdam, The Netherlands, June 2009), H. Weigand and S. Brinkkemper, Eds., vol. 479 of *ceur-ws*.

- SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. [117] Business Modeling for Service Engineering: Toward an Integrated Procedure Model. In *Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE'2009), Boston, Massachusetts, USA, July 1-3, 2009* (Boston, MA, USA, July 1-3, 2009), pp. 322–327.

- SCHEITHAUER, G., VOIGT, K., BICER, V., HEINRICH, M., STRUNK, A., AND WINKLER, M. [121] ISE Workbench: Integrated Service Engineering. Business Process Management Conference Demo Track, September 2009.

- KETT, H., SCHEITHAUER, G., WEINER, N., AND WEISBECKER, A. [61] Integrated Service Engineering (ISE) for Service Ecosystems: An Interdisciplinary Methodology for the Internet of Services. In *Proceedings of eChallenges e-2009 Conference* (Istanbul, Turkey, October 2009), P. Cunningham and M. Cunningham, Eds., IIMC International Information Management Corporation.

- SCHEITHAUER, G., VOIGT, K., BICER, V., HEINRICH, M., STRUNK, A., AND WINKLER, M. [120] Integrated Service Engineering Workbench: Service Engineering for Digital Ecosystems. In *Proceedings of the International ACM Conference on Management of Emergent Digital Ecosystems (MEDES)* (Lyon, France, October 2009), pp. 446–449.

2010

- SCHEITHAUER, G., AND WIRTZ, G. [124] Business Modeling for Service Descriptions: A Meta Model and a UML Profile. In *APCCM*

2010, 7th Asia-Pacific Conference on Conceptual Modelling (Brisbane, Australia, January 18-21, 2010).

- SCHEITHAUER, G., KETT, H., KAISER, J., HACKNER, S., HU, H., AND WIRTZ, G. [119] Business Modeling for Service Engineering: A Case Study in the IT Outsourcing Domain. In *SAC 2010, 25th Symposium On Applied Computing, Enterprise Engineering Track* (Sierre, Switzerland, March 22-26, 2010).

- HU, H., SCHEITHAUER, G., AND WIRTZ, G. [51] ISE – Integrated Service Engineering: Applying an Architecture for Model to Model Transformations. In *2010 International Conference on Software Engineering and Knowledge Engineering (SEKE'10)* (Redwood Shores, California, USA, July 1-3, 2010).

- BICER, V.; BOGERT, S.; WINKLER, M.; SCHEITHAUER, G.; VOIGT, K.; CARDOSO, J. AND AITENBICHLER, E. [21] Modeling Services using ISE Framework: Foundations and Extensions. In *Modern Software Engineering Concepts and Practices: Advanced Approaches* (Chapter 6, IGI Global, 2010).

2. Preliminaries

Following the general introduction and objective of this work, this chapter presents preliminary work that is important for understanding the motivation and the solution for a service description method. In contrast, section 9 discusses work that is related to the solution presented in this work. The first section addresses basic concepts and technologies, which includes method engineering, modeling notation development as well as frameworks in the area of business modeling. The then following section introduces the concept of service ecosystems and offers definitions, a link to Service-oriented Architectures and business networks, actors and life cycles, business models, and challenges. The final section shows different views on service descriptions.

2.1. Basic Concepts and Technologies

2.1.1. Method Engineering

Method engineering is a theory about the development of methods in the IT domain. Such methods comprise existing experience and knowledge in a domain and offer a structured approach in terms of guidance as well as documentation. Method engineering supports the formalization of this knowledge and to share it among practitioners.

According to Gutzwiller [46], a method embodies (1) meta models for (2) result document specification, (3) activities to guide the modeling process, (4) role definitions, (5) tools specification, and (6) techniques (cf. figure 2.1). *Result Documents* embody necessary knowledge gathered throughout an IT process. This includes, e.g., a requirement document or an

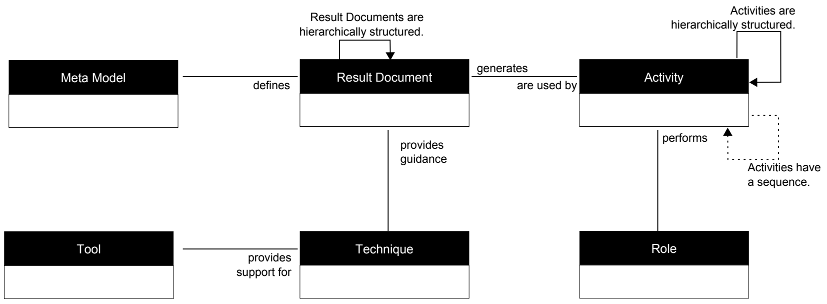


Figure 2.1.: Method Engineering (cf. [46]).

architecture document. Result documents can be decomposed into sub-documents. *Meta models* define result documents by specifying a knowledge structure by means of concepts and their relationships. *Activities* comprise best practices about which steps are to be performed in order to generate result documents. During the performance of activities semi-final result documents may be used as input. Activities may be disaggregated into sub-activities. Furthermore, sequences keep activities linked to each other. *Roles* acknowledge the fact that people with different skills are needed at certain stages in a method. A role defines a specific set of human skills which are needed for an activity. *Techniques* describe helpful theories to complete result documents, which include, e.g., data modeling, workflow modeling, and interviews, just to name a few. *Tools* lastly, provide support for techniques.

2.1.2. Developing novel Modeling Notations with UML Profiles

The Unified Modeling Language (UML) [87] is an accepted and well-known graphical language. Originally it aims at object-oriented design, but is not limited to it. UML Profile is part of the UML specification and offers a standard way to customize UML diagrams to cover domain-specific semantics. Standard UML and these *profiles* form the basis for a domain-specific modeling notation. This enables practitioners, who are already familiar with UML, to model specific domains. Well-known UML Profiles include SysML [94] and SoaML [91]. SysML is a notation to specify, analyze, and design IT systems. SoaML, on the other hand, is a notation useful to design a service-oriented architecture.

Giachetti et al. [40] provide a UML Profile generation process to transform domain-specific languages (DSL) into UML Profiles, which consists of three main steps:

1. Mapping between DSL and UML Meta Model – *Establish a mapping between elements of the DSL and UML meta model.*
2. Meta Model Comparison – *Identification of differences between DSL meta model and UML meta model.*
3. Transformation – *Setup of transformation rules and generation of a valid UML profile.*

2.1.3. Framework for Information Systems Architecture

The Framework for Information Systems Architecture (Zachman Framework) [144] provides a taxonomy to relate real world concepts to Enterprise Architecture [128]. Zachman describes Enterprise Architecture as means to flexibly react to business changes and to manage the varied resources of an enterprise. The Zachman Framework embodies vital artifacts to describe, create, operate, and change an object. The term *Object* is used consciously, since it may relate to practically anything, e.g., an enterprise, a project, or a solution.

Information systems' complexity increased exceedingly for two reasons [144]. Firstly, hardware and software improved with respect to price, availability, and capacity. Secondly, information systems were not only programmed to compute, but designed and implemented to support business operations. Zachman concluded that an architecture framework for

	Data	**Process**	**Location**	**People**	**Time**	**Motivation**
Scope Layer	List of important Entities	List of important Processes	List of important Locations	List of important Organizations	List of important Events	List of important Goals and Strategies
Business Layer	Semantic Model	Business Process Model	Business Logistic Model	Workflow Model	Master Schedule	Business Plan
System Layer	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technical Layer	Physical Data Model	System Design	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Component Layer	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification
Operations Layer	DATA	FUNCTION	NETWORK	ORGA-NIZATION	SCHEDULE	STRATEGY

Figure 2.2.: A Framework for Information Systems Architecture [144]

information systems is required to integrate and to interface the different enterprise artifacts.

The initial framework was developed by Zachman in 1987. In his article [144], he described how houses and air planes are build and who is involved in which part of the building process. Finally, he draws an analogy toward enterprise architecture. In 1992 Sowa and Zachman offered an extension and integrity rules [128]. In 1997, Inmon et al. [54] improved the framework rules. In 2008, Zachman [145] defined the scope and the limitation of the framework. Zachman limits the framework to a structure for describing an enterprise. It does not offer a methodology which guides the process of describing enterprises [145]. This includes that no concrete models are proposed, whether a top-down or bottom-up approach is favored, and how flexible the relations between descriptions and perspectives are.

The Zachman Framework distinguishes between six perspectives and six descriptions which are orthogonal to each other. Figure 2.2 depicts the framework with its two dimensions. Each column of the matrix offers a basic model for the description in question from a certain perspective. It is important to note that the framework does not specify the order of descriptions. Each intersection is a placeholder for a basic notation which satisfies a columns' basic model.

The six different perspectives are organized into corresponding layers [128]. It is important to note that the various perspectives are different with respect to nature, content, and semantics and not only in their detail level [144]. The *scope layer* represents the planner's perspective. The purpose of this layer is to identify *"... the size, shape, spatial relationships, and final purpose of the final structure."* [128] and thus, the scope. On this basis an owner of an enterprise decides whether to invest in the architecture. The *business layer* symbolizes the owner's perspective. Architects describe the requirements from the owner's perspective, whereas the intention is to *"... enable the owner to agree or disagree with the ..."* [144] description. The *system layer* corresponds to the designer's perspective. The purpose of this layer is to transform the enterprise model artifacts into detailed specifications. The owner can use these specifications to negotiate with builders to implement the system. The *technology layer* represents the builder's perspective. The rationale of this layer is that the detailed specifications must be adapted into builder's plans to take into account the *"... constraints, of tools, technology, and materials."* [128]. The *component layer* symbolizes the perspective of a sub-contractor. Builder's plans are translated into shop plans. Shop plans "... specify details of parts or subsections ..." [128] of builder's plans. The *operations layer* represents the system itself.

The six descriptions depict an enterprise from different angles. Though each of them is unique and addresses a different purpose, they relate to each other [144]. Descriptions are the answers to the basic questions: What (Data Description), How (Process Description), Where (Location Description), Who (People Description), When (Time Description), and Why (Motivation Description). It is important to note, that for each description exists a set of terms (description model) which are valid for all perspectives. Nonetheless, these terms differ essentially in *semantics* for each perspective. The *data description*'s model consists of entities and relationships between entities. The basic intention is to identify enterprises' inventory. For example, on the business model layer, entity refers to business entities, such as customer, and relationship refers to business relationships, such as company A is a supplier for company B. On the system layer, however, an entity refers to a data record, and a relationship refers to a data relationship. The *process description*'s model embodies processes and arguments (input and output to processes). The purpose is to make out enterprises' processes and business functions. For example, on the scope layer a process describes a highly aggregated business function. Arguments are not defined here. On the technology layer, a process refers to a computer procedure, where data types serve as arguments. The *location description*'s model uses the concepts of locations and connections in order to discover enterprises' networks. For example, on the system layer, a location refers to the physical location of a storage unit and a processor, and a connection refers to phone connection or email connection. The *people description*'s model is that of roles and work. The description's intention is the *"... allocation of work and the structure of authority and responsibility."* [128]. For example, on the business layer a role refers to an organizational unit, and work refers to product. On the component layer, role refers to a technical identifier, and work to a system transaction. The

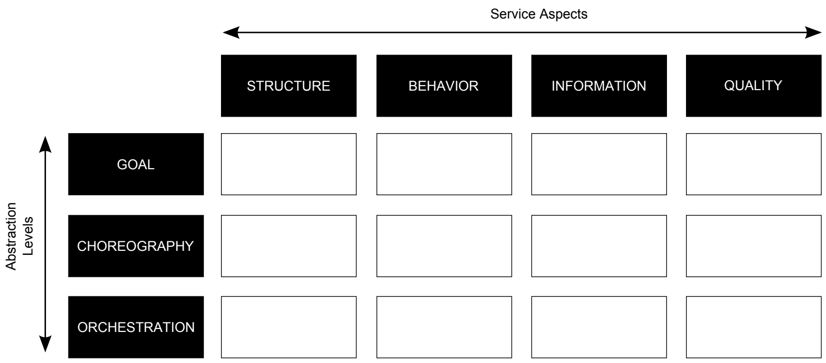


Figure 2.3.: A conceptual Framework for Service Modeling and Refinement (cf. [108])

time description's model embodies event and cycle. The description's intention is to *"... produce a schedule of events and states that maximizes the utilization of available resources while at the same time satisfying the external commitment."* [128]. For example, on the system layer, event refers to an event in an information system, and cycle refers to process cycle. The *motivation description*'s model uses the concepts of ends and means. The motivation description's intention is to describe the motive of an enterprise, where ends equal objectives and means equal strategies. For example, on the business layer, end refers to business objectives, and means refers to business strategies. On the technology layer, end refers to a technical condition, and means refers to a technical action.

2.1.4. Conceptual Service Modeling Framework

Quartel et al. [108] provide a general definition for a complete service concept, which states crucial service aspects to be modeled during the service engineering process. Such a service concept should be applicable for building complex services as well as service discovery.

In order to do so, Quartel et al. describe a framework for conceptual service modeling (COSMO) that is shown in figure 2.3. This framework is a step toward a generic service concept definition, which provides consistency between different service aspects. Furthermore, the authors conclude that the framework also serves as a common semantical basis for heterogeneous modeling notations.

Quartel et al. approach the framework's development with an analysis of available service definitions from diverse points of views, which include: (1) service as interaction, (2) service as capability, (3) service as operation, (4) service as application, and (5) service as feature. Following this, they assimilate generic service properties from these definitions. Generic service properties comprise that services involve *interaction* between two or more agents, that services provide *added value* for service consumers, and that several services can be *composed* into one service as well as *decomposed* into services.

The framework comprises two orthogonal dimensions: service aspects and level of abstractions (cf. figure 2.3). Each intersection is a placeholder for models or implementation languages. This framework supports to establish coherence between different service description artifacts.

Abstraction Levels: Quartel et al. distinguish between three different levels for service modeling: single interaction, choreography, and orchestration. The *single interaction* level focuses on a single interaction between service providers and service consumers. From the perspective of service providers it defines a service's capability (or value). From a service consumer perspective, it defines a need (or goal).

The *choreography* level concentrates on services' external behavior in that it refines a single interaction into multiple interactions between consumer and provider. Scarcely, services' value is accessible by means of a single interaction, e.g., accessing stock market data. Rather, consumers and providers must interact frequently to fulfill a service contract, e.g., a flight booking service.

The *orchestration* level applies only to service providers. It shows the combination of other services and their ordering in order to implement services' expected behavior.

Service Aspects: Service aspects include: structure, behavior, information, and quality. The *structure* aspect focuses on the modeling of interacting systems and their services. This includes service functional interfaces as well as their ports. The *behavior* aspect concerns about system activities, their relations, and their ordering in time. Behavior describes the single interaction as well as the choreography between consumers and providers. Additionally, it describes orchestrations of other services as well. The *information* aspect targets the modeling of domain entities. These entities are the basis of messages which are exchanged between service providers and consumers. Value is created by exchanging messages. The *quality* aspect comprises the modeling of services' non-functional properties.

Available Notations: Quartel et al. understand COSMO as a common semantic meta model for notations and their different modeling purposes. Modeling purposes relate to the aforementioned service aspects. The authors distinguish between three language categories: (1) design and specification languages, (2) analysis languages, and (3) implementation languages. Unified Modeling Language (UML) [87], Interaction System Design Language (ISDL), and Business Process Modeling Notation (BPMN) [83] fall into the design and specification language category. Analysis languages comprise Petri Nets and OWL-DL. Business Process Modeling Language (BPEL) [10] and Web Service Description Language (WSDL) [26] belong to the implementation language category.

2.1.5. Open-EDI Reference Model

The Open-EDI reference model [55], as shown in figure 2.4a distinguishes between the Business Operation View (BOV) and the Functional Service

View (FSV). BOV comprises business data semantics as well as business transaction rules, such as agreements and obligations between business partners. FSV, on the other hand, focuses on information technology which includes interfaces, functional capabilities, and protocols.

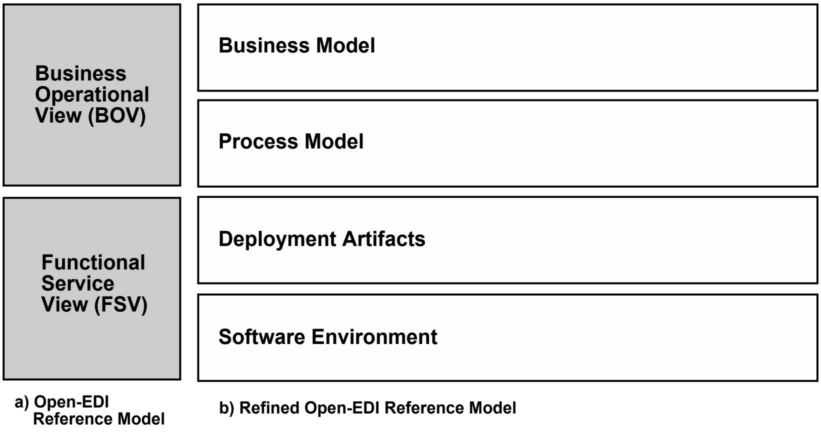

Business Operational View (BOV)
Functional Service View (FSV)
a) Open-EDI Reference Model
Business Model
Process Model
Deployment Artifacts
Software Environment
b) Refined Open-EDI Reference Model

Figure 2.4.: Refined Open-EDI Reference Model (cf. [33, 55]).

Dorn et al. [33] add subtle refinements to the Open-EDI reference model. Figure 2.4b discloses how they refine BOV into a business model and a process model. Business models express value exchanges between different actors and business analysis and concentrate on *what* needs to be done [117]. Process models, on the other hand, represent *how* each actor realizes such value exchanges [117]. Likewise, Dorn et al. refine FSV into deployment artifacts and software environments. Deployment artifacts address implementations of business processes with technical specifications. Software environments describe runtimes to execute technical artifacts. This refined model serves as a classification system for concepts and modeling notations as well as to define means to bridge gaps between different layers.

It is possible to use this refined model for the following purposes. Firstly and secondly, it offers abstraction layers as well as discreet phases for linking business and IT. Thirdly, semantics and intention can be defined for each of the four layers. Furthermore, the layers may group different stakeholders. Lastly, the reference model enables classifying appropriate modeling notations as well as realization languages.

Table 2.1.: Classification of Process Definition Concepts (cf. [33])

Business Model	**Process Model**	**Deployment Artifacts**
BMO [98]	BPMN [93]	BPEL [10]
e^3 Value [43]	EPC [60]	ebXML [34]

Table 2.1 shows an example for the application of this reference model that Dorn et al. [33] provide. They classify available concepts for defining processes according to the reference model's layers. The authors assort the Business Model Ontology [98] and the e^3 Value Ontology [43] (cf. section 4.1) for the business model layer. The process layer holds notations such as the Business Process Modeling Notation (BPMN) [93] and Event-driven Process Chains (EPC) [60], and the deployment artifacts, finally, depicts process realization languages including Business Process Execution Language (BPEL) [10] and ebXML [34].

2.2. Service Ecosystems

This section introduces ideas and concepts that are related with service ecosystems. It is important to note that the term *Service Ecosystem* spans ideas that are borrowed from other approaches with varying terminology. In this work, the terms *service systems* [129], *Internet of Services (IoS)* [114], and *Digital Ecosystems* are used synonymously to service ecosystems.

Web services [138] for heterogeneous application integration and communication between companies gained popularity during the last years

[123]. Recently, companies, such as Amazon.com, acknowledged web services beyond integration as a means to create value for customers. In consequence, the service ecosystem concept gained momentum. The vision of service ecosystems is an evolution of service orientation and takes services from merely integration purposes to the next level by making them available as tradable products on service delivery platforms [18].

This section applies the Open-EDI reference model [55] that differentiates between a business operational view and a functional service view in order to categorize the different perspectives on service ecosystems.

The following subsection elaborates on service definitions. The then following subsections define service ecosystems as an evolution of service-oriented architectures and business networks along with general actors and five generic business models. Finally, impediments for service ecosystems will be explained.

2.2.1. Service Definitions

Before diving into important concepts around service ecosystems, definitions for services in the business operational and the functional services sense will be provided. Tables 2.2 and 2.3 give an overview of some service definitions. This work will not provide another definition for services. Rather, all definitions are valid and they together provide a better understanding of the matter.

Zeithaml and Bitner [146] define services as follows: *"... services are deeds, processes, and performance ..."*. Kotler [65], on the other hand, describes services as *"... any act or performance that one party can offer to another that is essentially intangible and does not result in the ownership of anything. Its production may or may not be tied to a physical product."*

Normann [78] says that *"Service depends on division of labour and effective co-creation of value, leading to complementary specialization and comparative advantage among participants."* Lovelock and Wright [71] define services as

Table 2.2.: Business Service Definitions.

Year	Author	Process / Act / Performance / Activity	Intangible	Provider - Consumer - Interaction	Division of Labor	Optional Physical Product	Solution / Value Co-creation	No Ownership
2000	Zeithaml and Bitner [146]	x						
2000	Kotler [65]	x	x			x		x
2001	Normann [78]			x	x		x	
2002	Lovelock and Wright [71]	x		x			x	
2004	Baida et al. [16]	x	x	x				
2006	Lusch and Vargo [73]	x		x	x			
2007	Grönroos [45]	x	x	x		x	x	

"... economic activities that create value and provide benefits for customers at specific times and places, as a result of bringing about a desired change in – or on behalf of – the recipient of the service."

Baida et al. [16], however, see services as *"... business activities that often result in intangible outcomes or benefits; they are offered by a service provider to its environment."* Lusch and Vargo [73] point to specialization with their service definition: *"Rather, we define services as the application of specialized competences (knowledge and skills) through deeds, processes, and performances for the benefit of another entity or the entity itself."*

Grönroos [45], finally, defines a services as *"... an activity or series of activities of a more or less intangible nature that normally, but not necessar-*

Table 2.3.: Technical Service Definitions.

Year	Author	Software System	Machine-machine interaction	Interface / Service Description
2004	W3C Working Group [138]	x	x	x
2006	OASIS [97]		x	x
2007	Studer et al. [130]			x

ily, take place in the interactions between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems."

The W3C Working Group [138] defines a *"... Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL).".*

The Organization for the Advancement of Structured Information Standards (OASIS) [97] describes a service as follows: *"A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description."*

Studet et al. [130] relate to semantic in technical services. They define *"Semantic Web Services employ Semantic Web Technology in the web service area and compromise service functionality, web service inputs and outputs, their precondition and effects which are all expressed in knowledge representing languages, such as ontologies."*

Table 2.4.: Characteristics of Service Ecosystems (cf. [136]).

Characteristics	**Traditional Business Approach**	**New Business Network Approach**
Products and Services	Relative simple, unbundled, and slowly delivered products and services	Relative complex, unbundled, and fast delivered products and services
Value Creation	Supply chains with long term connected relationships	Demand networks with quick connect and disconnect relationships
Coordination and Control	Hierarchical and central control and decision making	Network orchestration with distributed control and decision making
Information Sharing	Information sharing with direct business partners	Information sharing over and with network partners
Infrastructure	Actor performs with information silos and systems	Network platform with networked business operation system

2.2.2. Service Ecosystems as Internet Market Places

As aforementioned, it is possible to approach the concept service ecosystem from two different angles. In general, service ecosystems comprise two main concepts. Firstly, it is a business network architecture that defines companies' business models, intra-company value chains, and forms of interaction. Secondly, it utilizes service-oriented architectures for a technical realization. Moreover, it is a market place which shows how to trade services over the Internet.

From a **business-operational view**, service ecosystems are market places for trading services in the business sense and involve actors from different legal bodies. Service ecosystems bring together shared information, people, and technology [129]. They comprise service innovation, service design, service engineering, service marketing, and service provisioning [62]. These systems provide core services such as payment and monitoring, domain-specific services such as eco-value calculations [56], and complex services such as travel services. These services are leveraged by others to implement end-to-end business processes [125], which cross companies' borders, to create value for end-customers. Business models such as Business Webs [132] foster this idea of *coopetition*. Service trade involves the following steps: service discovery, service selection, service contracting, service consumption, monitoring, and profiling. During discovery and selection, service providers advertise their services toward potential consumers, whereas service consumers specify their service preferences toward providers. During service contracting, providers and consumers negotiate and finally agree on service levels (SLA) which are monitored (for billing & payment) throughout service consumption. In the event service levels are not met, compensations must be triggered. During service profiling, valuable information on services' performance is stored, which is gathered during consumption and monitoring. Van Heck and Vervest [136] describe characteristics for service ecosystems, which are summarized in table 2.4.

Three phases lead to service ecosystems [132]. During the first phase corporations were vertically integrated, value creation was supplier-driven, and resources were scarce and physical. The second phase brought the virtual corporation. Corporations became more decoupled from each other and outsourcing was popular. The value creation process was more customer-driven but still strongly reflected supply-driven aspects. The third phase introduced service ecosystems. Within service ecosystems, value creation is totally customer-driven and service-enhanced, and resources become more digital and less physical.

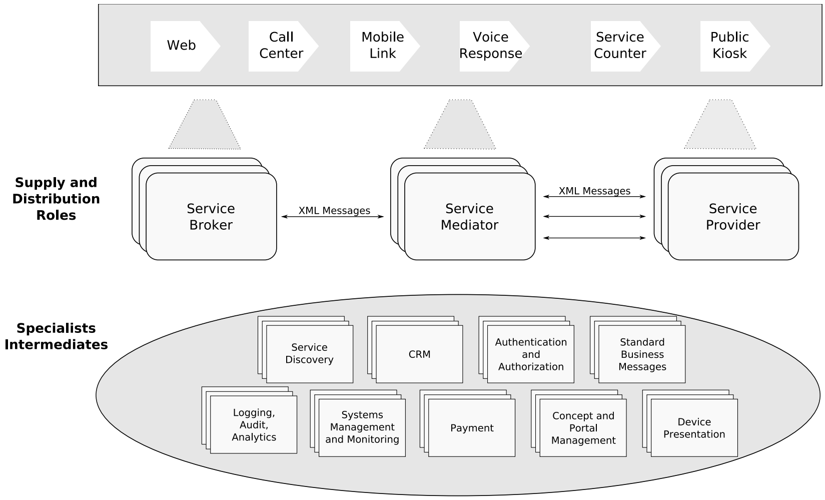

Web
Call Center
Mobile Link
Voice Response
Service Counter
Public Kiosk
Supply and Distribution Roles
Service Broker
XML Messages
Service Mediator
XML Messages
Service Provider
Specialists Intermediates
Service Discovery
CRM
Authentication and Authorization
Standard Business Messages
Logging, Audit, Analytics
Systems Management and Monitoring
Payment
Concept and Portal Management
Device Presentation

Figure 2.5.: Top-level Architecture of a Service Ecosystem [18]

From a **functional service view**, service ecosystems refer to a logical web service collection [18], with more than one service provider. Barros and Dumas [18] see Web Service Ecosystems (WSE) as an evolution of Service-oriented Architecture (SoA) (cf. figure 2.5). The authors describe SoA as a novel paradigm in order to combine legacy applications, automate business processes as well as foster technical integration between different legal bodies. Contrary to implementing business logic into hard-wired applications, software developers define technical services as fine-grained, reusable, loosely coupled functionality, which in turn can be wired according to actual business requirements. Recent developments show that once companies adapt to this paradigm, services are treated as valuable assets which can be exposed to other companies. Companies may offer and procure, and hence, trade these assets beyond organizational boundaries.

Furthermore, there exists a **combined view** on service ecosystems. Chang and West [25], for example, who relate to the term Digital Ecosystems (DE), address the way of how actors interact with each other. The authors ascribe that this new development will shift the business to business interaction from *"...centralized, distributed or hybrid models into an open, flexible, domain cluster, demand-driven, interactive environment."* Briscoe and de Wilde [23] see potential for optimization in the current way companies conduct their business in that they relate biological ecosystems to business ecosystems. Furthermore, the authors attribute the Internet as an enabler for this optimization. Janiesch et al. [56] see service networks where a service is provided by different actors. The authors acknowledge that realization of such networks involves business services as well as technical details involving web service technology. Internet of services' main aims is foster service trade, ability to bundle services, which in turn open new markets for small and medium enterprises, so the authors say. Tapscott et al. [132] speak of service ecosystems in terms of *"... distinct systems of suppliers, distributors, service providers, platform providers, and customers that use the internet for their primary business communications and transactions."*

2.2.3. Roles in Service Ecosystems

Following the discussion of different views on service ecosystems this subsection outlines diverse players in service trade. Existing literature reviews in the area of service ecosystems [18], business value webs [132] and Internet of Services [56] find evidence for different roles for actors. All the same, actors may play more than one role in service trade. Table 2.5 gives an overview of different actor roles.

Tapscott et al. [132] distinguish between consumer, context provider, content provider, commerce service provider, and infrastructure provider. Consumers demand and consume goods and services. Context providers provide a single face to the customer. They lead the process of value creation, in terms of orchestrating service ecosystems in such a way that value meets consumer needs. They also provide a set of rules for each

Table 2.5.: Possible Roles in Service Ecosystems.

Author	**Consumer**	**Provider**
Tapscott et al. [132]	Service Consumer	Context Provider
		Content Provider
		Commerce Service Provider
		Infrastructure Provider
Papazoglou [105]	Service Client	Service Provider
		Service Aggregator
		Service Operator
		Market Maker
Barros and Dumas [18]	Service Consumer	Service Provider
		Mediator
		Broker

stakeholder in service ecosystems. Content providers are the main value contributors. They actually design, create, and deliver goods and services to meet customer needs. Commerce service providers offer services with a cross sectional character. These services include financial management, security, logistics, and monitoring for example. They enable the stream of value creation in service ecosystems. Infrastructure providers offer services in terms of communication platforms, computing, buildings, networks, facilities, and roads.

Barros and Dumas [18] on the other hand, identify next to service consumers three different roles for actors in service ecosystems. Service providers provide services in the first place. Service brokers offer services from different providers. Their business model is to bring providers and consumers together, or enhance services with delivery functions for convenient service provisioning. Service mediators, on the other hand, generate value by customizing provider's standard services toward consumer's needs.

Papazoglou [105] offers next to a service client four different roles in his work about an extended SoA. Service providers offer services to others. A service aggregator, on the other hand, is a service provider who aggregates services from different providers in order meet specific needs and hence, generate added value for consumers. According to Papazoglou, aggregators are responsible to coordinate, to monitor, and to ensure integrity of service compositions. Service operators are responsible for operating trans-sectional functions such as service deployment, platform, and application monitoring. The market maker, finally, is an association of interacting organizations. Market makers provide a means to bring clients and providers together.

Even though there exist different notions for actors, this work will not offer another actor definition. Rather, the definitions will be used simultaneously.

2.2.4. Business Models for Service Ecosystems

While the previous subsections defined service ecosystems, this subsection presents possible settings and interactions of actors in terms of business models. Tapscott et al. [132] introduce five generic business model types. Each type is defined by a definition, a value proposition, different subtypes, market environments, and key success factors.

The five generic business models can be *categorized* along two dimensions: Economic control and value integration as shown in figure 2.6. Economic control refers to how hierarchical or self- organizing service ecosystems are organized. Hierarchical service ecosystems have an organizer who manages the offer of prices, the value proposition, and the stream of business. Although Amazon.com offers a great selection of goods from different suppliers, Amazon.com sets the price for each good. Self-organizing service ecosystems however, define prices and value for goods and services on the basis of market dynamics. For example, in an eBay auction the value of a good is determined by the seller and the bid-

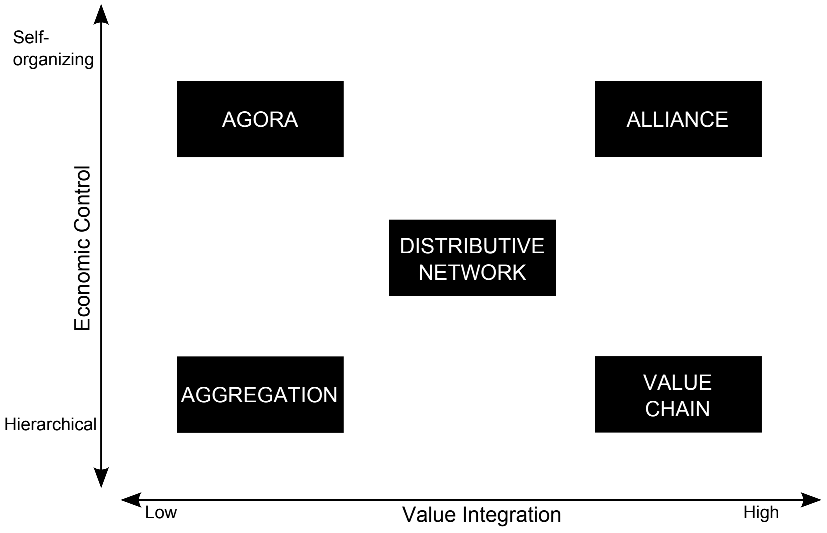


Figure 2.6.: Service Ecosystems Categorization (cf. [132])

der, not by eBay itself. Value integration, on the other hand, refers to the level of how integrated the value contribution from each stakeholder is. Tapscott et al. define value as *"... the benefit that a user gains from a good or a service"*. Some service ecosystems have a high level of value integration. General Motors for example, achieves a high level of value integration by combining the contributions from many suppliers and integrating them in a further process into a final product. Figure 2.6 shows a classification for business value network types along the two dimensions. The following subsections address each business value network type in detail.

AGORA The Agora business model supports the trade of goods and services between providers and consumers in terms of discovering prices for goods and services in real-time, either in a one-to-one negotiation or in a public auction as shown in figure 2.7. Agoras are not limited to certain kinds of goods or services. Providers and consumers are in the position to access Agoras without difficulty to offer or to buy goods and services.

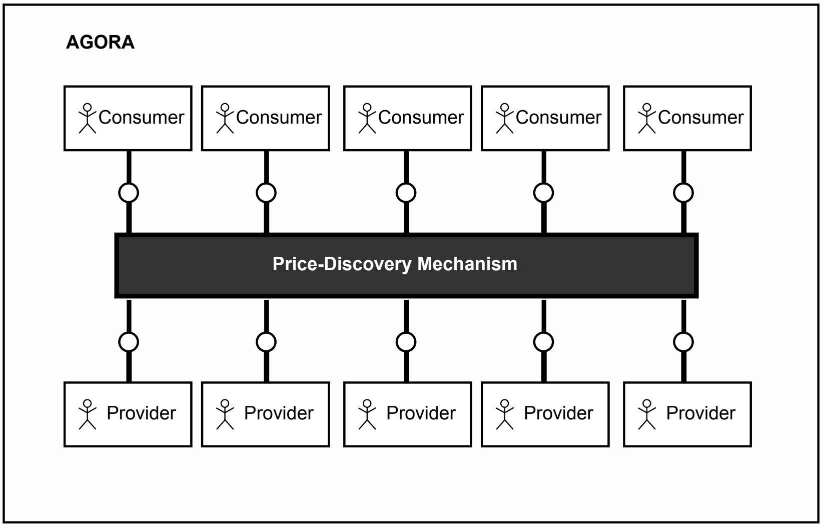


Figure 2.7.: Agora (cf. [132])

Agoras' main theme is dynamic pricing almost without any transaction costs. Tapscott et al. define an Agora as follows: *"Liquidity, the ease of converting assets into cash, is the core value proposition of Agoras. Agoras achieve liquidity by matching buyers and sellers and by facilitation of price discovery, whereby buyers and sellers cooperate and compete to arrive at a mutually acceptable exchange"*.

Value Proposition refers to the value of service ecosystem actors. Agoras value proposition comprises liquidity and price discovery. Liquidity describes the possibility of converting goods and services into cash. For example, eBay users are in the position to offer any good at eBay's website for potential buyers. Price discovery is not based on difficult negotiations between buyers and sellers but in a dynamic fashion. This leads to less transaction costs for buyers and sellers. Tapscott et al. [132, p. 40] argue that *"Agoras tend to prevail where the transaction costs of negotiating are lower than the range of uncertainty about the final price"*.

Agoras appear in four different **forms**. In an open market, there is exactly one provider and one consumer. In general the products are unique like in the real-estate market or in the job market (e.g. Monster.com). The price-discovery strategy is a one-to-one negotiation. In a sell-side auction, there is only one provider and many consumers. The products are commercial and unique goods (e.g. eBay). The price-discovering strategies include for example the English auction model and the Dutch auction model. In a buyside auction are many providers and one consumer. This form of Agora appears in consumer's markets, where the providers are in a high competition. The price-discovering strategy is the request for quotations (RFQ). Exchanges are the most complex form of an Agora. Many providers and many consumers interact with each other. Products are commodity goods (e.g. NYSE). Providers and consumers might change roles often. Price-discovery strategy includes a variety of auction models and fast-pacing bid-and-ask mechanisms.

Agora business models appear in three **market environments** [132]: Business-to-business, business-to-consumer, and consumer-to-consumer service ecosystems.

To successfully develop or maintain an Agora, **eight success factors** must be addressed [132]. The price-discovery strategy must be very dynamic, user-centric, and frictionless, to decrease transaction costs for consumers and providers. Agoras build on a critical mass of participants to enable fully functional price-discovery between consumers and providers. It is necessary to continually improve and align Agoras to meet participants' needs. In general, Agoras are no level playing fields. Their rules support insiders to some extent. Thus, insider advantage should be considered as a design decision. Monitoring the behavior of participants is crucial to improve the service ecosystem. Agoras need to allow brokers and mediators for matching trades between consumers and providers. New opportunities will be created by them. The last success factor is about building and maintaining trust, privacy, and regulatory issues.

AGGREGATION Figure 2.8 shows an Aggregator as an intermediary between providers and consumers. For each Aggregator there exists a leader, mostly a company which takes over the role of a context provider. This leading company in an Aggregator is the main contributor of the value proposition such as selection, fulfillment, pricing, and market segmentation. An example is Amazon.com which aggregates products in categories from different providers.

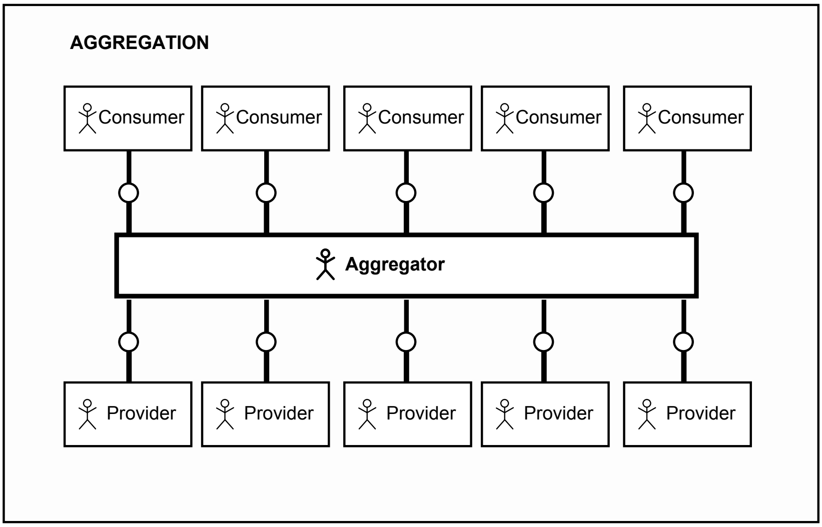


Figure 2.8.: Aggregation (cf. [132])

Aggregations **value proposition** comprises selection, organization, price, convenience, matching, and fulfillment. Selection refers to bundling, grouping, and packaging cross-vendor products. Organization is the process which determines the aggregation of services around any given user scenario. Amazon.com for example, categorizes products into functional categories as well as into special listings, such as gifts and bargains. Aggregations are in a position to leverage reduced transition costs, either to reduce prices for goods and services, or to cover additional surplus value. Convenience for consumers is created by the value-added service the Aggregation offers. Amazon.com's website for example, offers secure payment, free delivery of purchased goods, and powerful search facilities. Aggregations offer unique possibilities to match individual consumer needs. Amazon.com tracks consumers shopping behavior and derives product proposals for each individual consumer. Aggregators target fulfillment to better match needs of consumers. Fulfillment addresses higher margins, sophisticated demand management, order stratification, and efficiency.

Aggregation appears in **six different forms**. Superaggregations leverage scale efficiency and high return on investments by creating selections across traditional product categories. E-sources gather information about risky buying decisions. Then, consumers are in the position to leverage this information where in general only specialists would have access to (e.g. E*Trade). On the one hand, E-brokers offer producers to aggregate demand from many small consumers. On the other hand, E-brokers offer consumers to aggregate supply from many providers to meet consumer needs (e.g. Chemdex). Integrators join previously separated purchase processes for the convenience of consumers (e.g. Streamline.com). Industry hubs offer aggregated industry information and services to consumers as well as business transactions (e.g. Ariba). Many consumers are attracted by portals where they can freely aggregate content and services in their own manner and preferences. Providers can advertise their products within consumer portals and are charged when consumers follow their links from the portal to suppliers' goods and services (e.g. Yahoo!).

Aggregation service ecosystems appear in two market **environments**: Business-to-business, and business-to-consumer market environment.

To successfully develop or maintain an Aggregator **five success factors** must be addressed. Aggregators must design the value propositions around consumer needs. This is accomplished by tracking and understanding consumer behavior. Next to the importance of the value of offered goods and services, information about the offered goods and services must be managed as a separate value for consumers. Aggregators must enable content organization to increase consumer experience and control. Connected consumer communities are willing to contribute value to the business value network. Aggregators must leverage this value by offering the means for consumer networking. Aggregators must face the fulfillment challenge: Depending on the offered goods and services it is not enough selling services over the internet. Physical goods for example, still need to be delivered by mail. Aggregators must offer consumers a holistic way to meet their needs.

VALUE CHAIN Developing a highly integrated product by means of a managed process is the focus of a Value Chain, which is shown in figure 2.9. A Value Chain is led by one company, the context provider, which manages the value creation process. Tapscott et al. define Value Chains as follows: *"The value proposition of a Value Chain is the design and delivery of an integrated product or service that meets a specific set of customer needs. In a Value Chain, the context leader defines the goal and coordinates the value contributions of the various participants, controlling the design of the product and choreographing the key steps."*

Value Chains appear in two different **forms**. Routine production refers to mass production, product-centric, and make-and-sell paradigm. It is focused on the producing process which comprises the production itself, stocking, and selling products (e.g. GM, Ford). Shop production, however, is a more demand-driven form of a Value Chain. Shop productions develop unique solutions to address custom consumer demands, such as building and delivering a computer with a unique configuration matching the needs of a single consumer (e.g. Dell, Cisco). Without the consumer's request, this computer would never have been built. Shop production differs in three ways from routine production. First, production is never routine, but targeted to a specific consumer need. Second, shop productions

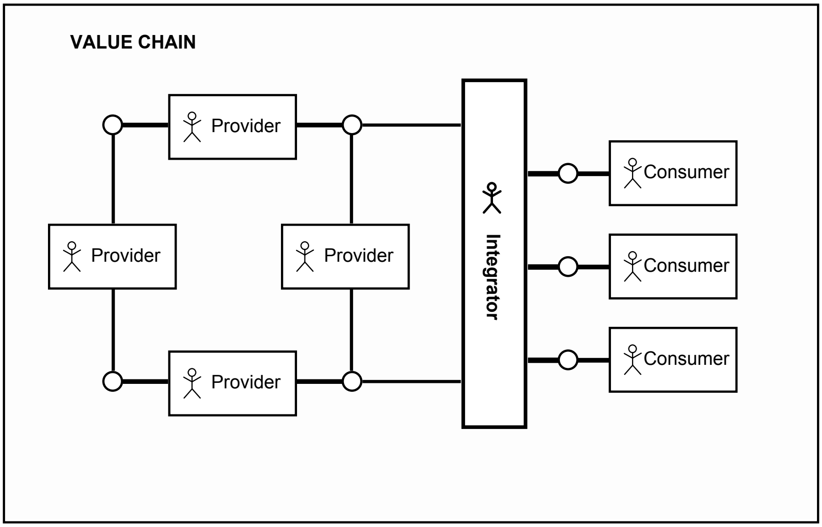


Figure 2.9.: Value Chain (cf. [132])

are totally driven by consumer demand. Third, consumers are part of the whole process of designing, building, and delivering the final solution.

Value chain service ecosystems appear in the business-to-business market environment.

To successfully develop or maintain a Value Chain **six success factors** must be addressed. Context providers must provide service-enhanced custom solutions. Consumers' surplus value around the good or service must be identified and turned into a fully integrated solution. Consumers must be included into the value chain. Context provider and content provider gain advantages by knowing and responding to consumer demand. The Value Chain must transform itself from a product-centric model to a consumer-centric model. Context providers must identify the most consumers' surplus value and tailor all activities around it. Relationship management must be addressed and continuously improved. This includes relationships to business value chain partners as well as consumers and potentially new partners. Knowledge must be shared within value chains.

ALLIANCE Within an Alliance, companies and individual people follow a common goal, where no one has the full control over it. The open source community is an example for an Alliance. Individual people and companies (e.g. IBM) contribute their time and money to software solutions which are freely available for everyone. There are, however, many different reasons behind this. Individuals, for example, just want to contribute time to open source projects to improve their skills in a certain technology. Companies might promote open source projects in order to complete their product portfolio with the final open source solution. A well known example is the Apache project, which develops an open source web server. Apache is the fundament for IBM's web application server Websphere. Tapscott et al. define Alliances as follows: *"The core value proposition of an Alliance is creative collaboration in aid of a goal that is shared across a community of contributors"*.

Alliance's **value proposition** comprises collaboration for a common good. Though content providers still follow their own goals, they drive and improve the common good in some ways. Consumers clearly benefit from the wide set of different goals. Linux for example is developed by very different people and companies and it in turn serves people and various companies. An individual person, like Linus Torvald for example, contributes to the Linux kernel, and a company like Red Hat, takes the value of the kernel and integrates it into a stable business operation system. However, it works the other way around as well. Linus Torvald may take advantage of the operation system from Red Hat, to further develop the kernel.

Alliances appear in five different **forms**. Social alliances do not target a higher purpose such as an integrated product. Rather, social alliances exist for their own sake (e.g. Dinner parties, chats, usenet, forums). Dis-

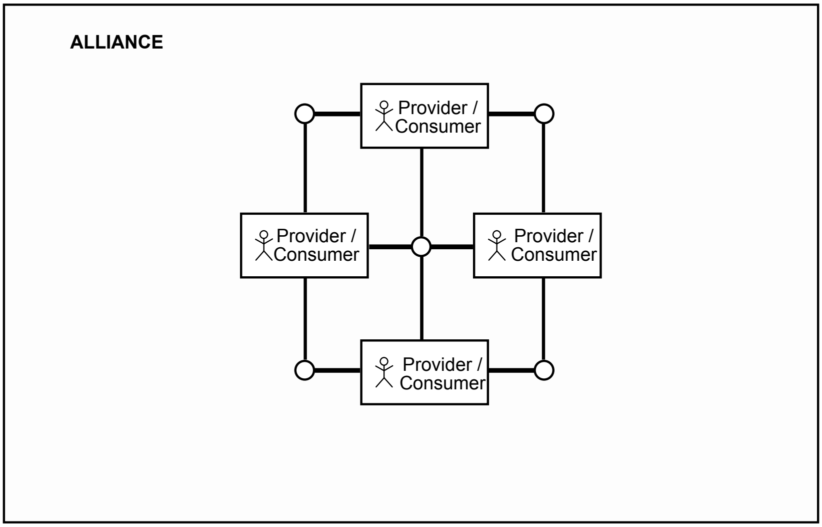


Figure 2.10.: Alliance (cf. [132])

cussion alliances address individual people who search for possibilities to learn on the basis of social interchange (e.g. GardenWeb, Harley Owner Group, Wikipedia). That means that people absorb knowledge from other people and redistribute their own knowledge to others. Help alliances address a certain scope of a problem. Within help alliances, people try to help each other. In general, open source project mailing lists are of great help for software users having problems with the software, since the developers of the software and experienced users participate on that mailing list. Especially experienced users are eager to support newcomers with problems they also faced in the beginning (e.g. SourceForge). In design collaborative alliances, participants also contribute to design, development, and distribution of a tangible product. The open source movement and global research projects follow the paradigm of a design collaborative alliance (e.g. Linux, Human Genome Project). Production alliances are very close to Value Chains with the exception of a missing global leader. Content providers are independent producers. However, they contribute their products to meet consumer needs (e.g. Deutsches Institut für Normung, Wintel (Microsoft and Intel)).

Alliance service ecosystems appear in three market environments: Business-to-business, business-to-consumer, and consumer-to-consumer market environment.

In order to successfully develop or maintain an Alliance **nine success factors** must be addressed. It is crucial, to assure content providers the capability to follow their own goals within an Alliance. Otherwise content providers will leave the network or disengage. Context providers must set a global vision for the Alliance. Power must be divided and shared within the network. Content providers must be treated on the basis of their contribution, rather than their entitlement. Alliances must support an environment where lively discussions and interactions are possible. Information boundaries must not exist between stakeholders. Content providers must be rewarded for their contribution, either with public recognition, or with financial rewards. Rules and the process of establishing rules within the network must be transparent for stakeholders. The last factor addresses the final outcome of the Alliance. It must be very modular, so that stakeholders are in the position to further contribute to a single module, and to absorb parts of it.

DISTRIBUTIVE NETWORK The distributing network model provides infrastructure services for other forms of service ecosystems (cf. figure 2.11). These infrastructure services comprise, for example, parcel services, communication, streets, electrical power grids, logistic services. Tapscott et al. define Distributive Networks as follows: *"The core value proposition of a Distributive Network is to facilitate the exchange and delivery of information, goods, and services"*.

Distributive Networks appear in three different **forms**. Slice-and-dice service ecosystems assemble and distribute a continuous stream for divis-

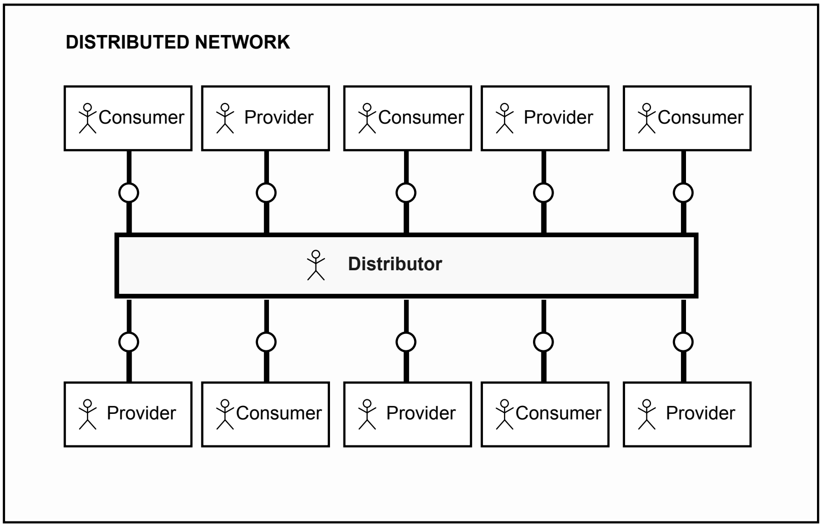


Figure 2.11.: Distributive Networks (cf. [132])

ible goods like power and natural gas. Store-and-forward service ecosystems comprise parcel and shipping services, as well as airline companies, highways, and the Internet. Within these networks, products get delivered intact and untouched. Park-and-lever service ecosystems include banks and insurance companies. Park-and-lever service ecosystems amass capital and lever the value by lending the capital to others for some interest.

Distributive networks appear in two market environments: Business-to-business, and business-to-consumer market environment.

To successfully develop or maintain a Distributive Network **four success factors** must be addressed. Service ecosystems enablement refers to the fact that other types of service ecosystems depend on services from Distributive Networks. Thus, Distributive Networks enable other service ecosystems by providing their services. It is important to choose between commodity infrastructure and value adding slices. Either, Distributive Networks only offer their core value to consumers, or they enhance their service by adding extra services to their offering to flexibly meet consumer needs. A hybrid approach is not an option. Event-driven responsiveness and optimization must be enabled to meet consumer demand. The last success factor is to maximize points of presence, which means that the more consumers and locations a Distribute network covers, the more value will be created.

2.2.5. Challenges for Service Ecosystems

While the previous text outlines service ecosystems as a means for trading services over the internet, the following paragraphs elaborate on current impediments for realizing successful IoS. Barros and Dumas [18] for example outline the following issues: service discovery, conversational multiparty interactions, and service mediation and adaption.

Barros and Dumas pinpoint that the current service discovery process depends on keyword-based searches. It is assumed that service providers as well as consumers use the same keywords for describing and discovering them. According to the authors, this works well in closed environments but not for multi-actor market places. Barros and Dumas advocate a combination of a free-text and ontology-based search. Schumacher [126] also encourages a XML-based service description. This work addresses this issue.

Additionally, while trading services over the Internet, interactions between actors will exceed traditional request-response patterns. In consequence, service ecosystems must support multiparty interactions as well as a formalization for defining them. Barros and Dumas foster two technical specifications for this: firstly, the Business Process Execution Language (BPEL) [10] and secondly, the Web Service Choreography Description Language (WS-CDL) [58].

Another challenge lies in integrating purchased services into companies' internal service systems. In the scope of service ecosystems, services may be used in contexts that were not initially considered by service providers, and hence, provide an interface that is inappropriate for others, including service mediators and brokers. This fact makes it necessary to mediate between services' given interface and an expected interface.

Luoma and Vahtera [72] address trust in business networks and argue that standards around rights and obligations must become available.

2.3. Different Views on Service Description

This section elaborates on different views on service description. Roman et al. [112] distinguish four distinct aspects for describing services that figure 2.12 shows.

The first view addresses service functional properties that declare a service's functionality or capability, whereas service functionality refers to the action a service actually performs [79]. The second view targets non-functional properties. Non-functional properties refer to constraints over the functionality of a service [101] and may include information about service quality levels, price information, or ratings. Choreography or service interaction is the third view. Service choreography refers to how service functionality is consumed [18] and includes message exchange patterns as well as the ordering of messages. The last view addresses service orchestration. Orchestration specifies how a service is composed from other services in order to achieve its functionality [130].

Next to these four views, service descriptions must be represented in a machine-processable and formal way. Firstly, the format must be formalized. This ensures a shared knowledge of the formal semantics of the service description [79]. Secondly, a persistence-format must be available to exchange service descriptions between different parties. Formalism is understood as an ontological source. An ontological source is an incomplete view on the world (or a view on a specific domain), and a knowledge representation. Thus, it forms a shared understanding between each party.

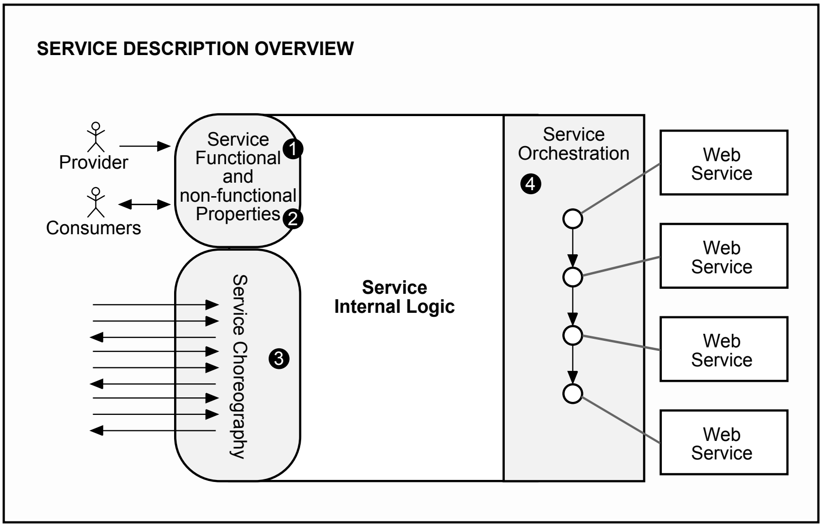


Figure 2.12.: Service Description Overview (cf. [112])

Ontological sources include dictionaries, thesaurus, ontologies, specifications, and standards [79]. Table 2.6 presents existing approaches, which address the four views along with their means of formalization.

The remainder of this section is organized as follows. While subsection 2.3.1 elaborates on functional service properties, subsection 2.3.2 addresses non functional service properties. Subsection 2.3.3 outlines service choreography and subsection 2.3.4 discusses service orchestration.

2.3.1. Functional Service Properties

As mentioned above, the functional behavior of a service depicts the action a service actually performs. Oaks et al. [79] present seven requirements for describing service functionality:

1. The ability to declare what action a service performs.
2. The ability to allow a capability to have different sets of inputs.

Table 2.6.: Service Description Overview.

Service Description View	**Model / Notation**	**Formalization**
Functional Properties	LARKS [131]	Case Frame
	EXPECT [42]	Structured Language
	WSMO [112]	Ontology
	OWL-S [74]	Ontology
	SWFS [20]	Ontology
	WSDL [27]	Normative Standard
Non-functional Properties	ORM [47]	Natural Language / Diagrams
	WSMO [112]	Ontology
	OWL-S [74]	Ontology
Choreography	WS-CDL [58]	Normative Standard
	BPEL [10]	Normative Standard
	WSMO [112]	Ontology
	BPMN 2.0 [31, 93]	Normative Standard
Orchestration	BPMN 2.0 [93]	Normative Standard
	BPEL [10]	Normative Standard
	UML [87]	Normative Standard
	WSMF [38]	Ontology

3. The ability to declare preconditions and effects in some named rule definitions.

4. The ability to describe objects that are not input but are used or affected by the capability.

5. The ability to refer to ontological descriptions of the terms used in the description and thus place the use of the terms in context.

6. The ability to make explicit the domain or context in which the service operates.

7. The ability to classify capabilities based on aspects of the description enabling exact or partial matches between required and provided capability descriptions.

The requirements mentioned above are interpreted as follows. Services provide a specific functionality in one or more domains and contexts. The functionality of services is composed of a set of relating capabilities, which address a specific subset of a service functionality. All capabilities together represent the whole functionality of a service. Each capability declares a set of inputs and outputs. Input refers to the type of information, the state of the information system (pre-condition), and the state of the domain and context of the information system (assumption), which is required in order to carry out the capability. Output refers to the type of information, the state of the information system (post-condition), and the state of the domain and context of the information system (effect), which is guaranteed after the execution of the capability [130]. The same approach can be used for defining service requirements in the service discovery process. Service requirements are then matched against service descriptions of available services. Five different degrees of matches are possible [103]:

- Fail: The service description does not match the service requirement.
- Subsumption match: The service description covers only some parts of the service requirement.
- Plugin match: The service description covers more than the service requirements.
- Intersection match: A part of the service description covers only some parts of service requirements.
- Exact match: The service description addresses the service requirements exactly.

This notion of service functionality requires a formal representation and shared understanding between different parties. A formal representation is provided by a formal language, such as LARKS [131], EXPECT [42], WSMO [112], OWL-S [74], and SWFS [20].

2.3.2. Non-Functional Properties of a Service

O'Sullivan et al. [101] refer to non-functional properties as constraints over the functionality of services. They argue that non-functional properties of services are an essential ingredient of a service description. Possible non-functional properties for services include price information or quality levels. Non-functional properties (1) improve service discovery (higher degree of expressiveness), (2) allow service substitution (better service comparison), (3) advance service composition, and (4) permit service management in terms of monitoring and controlling services [101]. Similar to the functional behavior, non-functional properties require a formal representation and a shared understanding between different parties. O'Sullivan et al. [100] offer a formal description for non-functional properties. They use Object Role Modeling (ORM) [47] for formalizing 14 different categories of non-functional properties. Their approach addresses services, which are consumed electronically (e-Services) as well as services which are consumed in a traditional manner.

Possible notions for representing non-functional properties include ORM [47], WSMO [112], and OWL-S [74].

2.3.3. Service Choreography

Service choreography is concerned with describing the externally visible behavior of services, as a set of message exchanges optionally following a Message Exchange Pattern (MEP), from the functionality consumer point of view. Barros and Dumas [18] depict that service interaction as a part of service description becomes crucial for long and multi-company running services. Single request-response exchange patterns are not likely for coarse-grained web services. More likely, services must support multi-party, and long running business processes [18]. Thus, a logical service description must support various interaction patterns to enable multi-party collaborative environments [19]. In [19], Barros et al. introduce a set of 12 different service interaction patterns. The service interaction patterns are classified with the following three dimensions:

- The maximum number of parties involved within a service interaction.
- The maximum number of interactions between two parties participating in one interaction.
- For two-way interactions whether the receiver of the *response* message is also the sender of the *request* message.

These patterns build a solid basis for describing the interaction with services. Several standards are available to express partly these interaction patterns. Web Service Choreography Description Language (WS-CDL) [58] is a W3C initiative to express web service choreography, which gains momentum for multi-party collaborating environments [18]. Barros et al. [19] offer a mapping from their service interaction patterns directly into BPEL [10]. Roman et al. [113] use abstract state machines to model the interaction between a service and a service consumer. Lastly, RosettaNet Partner Interface Protocols offer another possibility to declare interaction between different parties involved in an interaction.

2.3.4. Service Orchestration

Orchestration [113] deals with describing how a number of services, two or more, can be composed with the aim of achieving a common goal. The common goal represents the service consumer's formalization of a service requirement. In general, service providers should not expose the realization of a service (e.g. how a service is aggregated from other services) to service consumers. However, in the case that a service consumer's goal does not match to one available service description, but might be fulfilled by two or more services plugged together (Subsumption match), service consumers, service broker or service provider must be in the position to express this composition. Thus, service composition is an optional part for the formal service description. Service composition can be expressed with a variety of models. Available approaches are UML's Component Diagram, Business Process Execution Language (BPEL) [10], Web Service Modeling Framework (WSMF) [38], and BPMN [93].

This chapter presented preliminary work for understanding the motivation and the solution for a service description method. The first section addressed basic concepts and technologies, which included method engineering, modeling notation development as well as frameworks in the area of business modeling. The then following section introduced the concept of service ecosystems and offered definitions, a link to Service-oriented Architectures and business networks, actors and life cycles, business models, and challenges. The concluding section showed different views on service descriptions.

Part II.

Design

3. Describing Services for Service Ecosystems

This and the following three chapters outline the Service Description Method for Service Ecosystems (SDM4SE) [115]. Its intention is to provide a method suitable in order to define service descriptions in a business-oriented fashion and to transform them into technical specifications. While this chapter outlines the method as well as its motivation and cornerstones, chapter 4, 5, and 6 examine the method and its components in detail. The organization of this chapter is as follows: whereas section 3.1 introduces service description requirements and outlines the approach to cover these requirements, section 3.2 draws the method's outline with its components. Section 3.3 finally, introduces the *eco calculator* scenario as a running example for the next chapters.

3.1. Requirements and Approach

The next paragraph outlines three requirements of service descriptions. The then following paragraph elaborates on the approach taken that consists of a literature analysis, meta modeling, developing of modeling notations, and application of four different layers that bridge business and IT.

Service ecosystems are a new form of market places that enable service trade over the Internet [56]. One challenge in realizing service ecosystems lies in that physical goods are different to services. Contrary to goods, services are intangible, inseparable, variable, and perishable [66]. This means that services cannot be experienced like goods, which normally can be seen, tasted, felt, heard, or smelled (intangibility). Furthermore, the time of service production and consumption occur at the same time (inseparability). Also, service quality depends on external factors such as who services provides and consumes, as well as when and where services are consumed (variability). Lastly, services cannot be stored (perishability). In consequence, services can neither be easily *evaluated* in themselves nor be easily *compared* with other services [45]. Another challenge lies in supporting market actors. General market actors provide and consume services [18]. Specialized actors, on the other hand, offer market platforms, or support service trade with brokering and mediation [18]. Hence, market actors need a *formalization* that allows describing service offers and service needs [67]. Since service ecosystems utilize the Internet for service trade, it is important to translate service offerings and needs into Internet technology. This *business-IT-alignment* enables automatic service discovery, service monitoring and control [101]. In summary, a service description method for service ecosystems faces these three challenges: *service evaluation & comparison*, *formalization of service descriptions*, and *bridging business and IT*.

The method described in this work offers a means for service evaluation, description formalization, and an alignment between business requirements and Information Technology. This is accomplished with knowledge about service properties, the use of meta modeling as well as the development of novel modeling notations, an investigation of technical specifications for service descriptions, a mapping between service properties and technical specifications, and by establishing discreet layers according to the Open-EDI reference model [55]. In order to *evaluate* as well as to *compare* services, a service description method must offer instruments for declaring service properties that distinguishes between functional and non-functional properties. While functional properties refer to the action a service actually performs [79], non-functional properties refer to constraints over the functionality of a service [101], such as information about pricing, promotion, or productivity. This is accomplished with a careful analysis of service properties in available literature about business models and service marketing (cf. sections 4.1 & 5.1). For establishing a common way to describe services, the aforementioned service properties must follow a shared understanding of service descriptions. This is realized by using meta models for a *formalization* of functional and non-functional service properties (cf. sections 4.2 & 5.2), and an analysis of existing modeling notations (cf. section 3.2) as well as the development of novel modeling notations (cf. sections 4.3 & 5.3). An *alignment between business and IT* is impeded because of the ample technical specifications for describing services as well as the complexity involved in generating IT artifacts from business requirements. In order to foster an *alignment between business and IT*, the Open-EDI reference model [55] is utilized, which establishes four layers that connect business models with software environments, an analysis of existing technical specifications for service descriptions is carried out (cf. section 6.1), and a mapping between service properties and technical specifications is introduced (cf. section 6.2).

3.2. Service Description Method for Service Ecosystems

This section outlines the service description method that intends to remedy the issues involved with the development of service descriptions. The reference model, which is shown in figure 3.1, differentiates several service description modeling phases. It is based on the aforementioned Open-EDI reference model [55] and work of Dorn et al. [33]. The then following paragraphs introduce the reference model for service description and its components.

It is important to note that the refined Open-EDI reference model by Dorn et al. focuses mainly on process descriptions. Hence, the reference model for service descriptions adepts subtle changes as figure 3.1 shows. Scheithauer et al. [118] argue that service properties in the *Business Service Model* layer own a strategic semantics and take into account services'

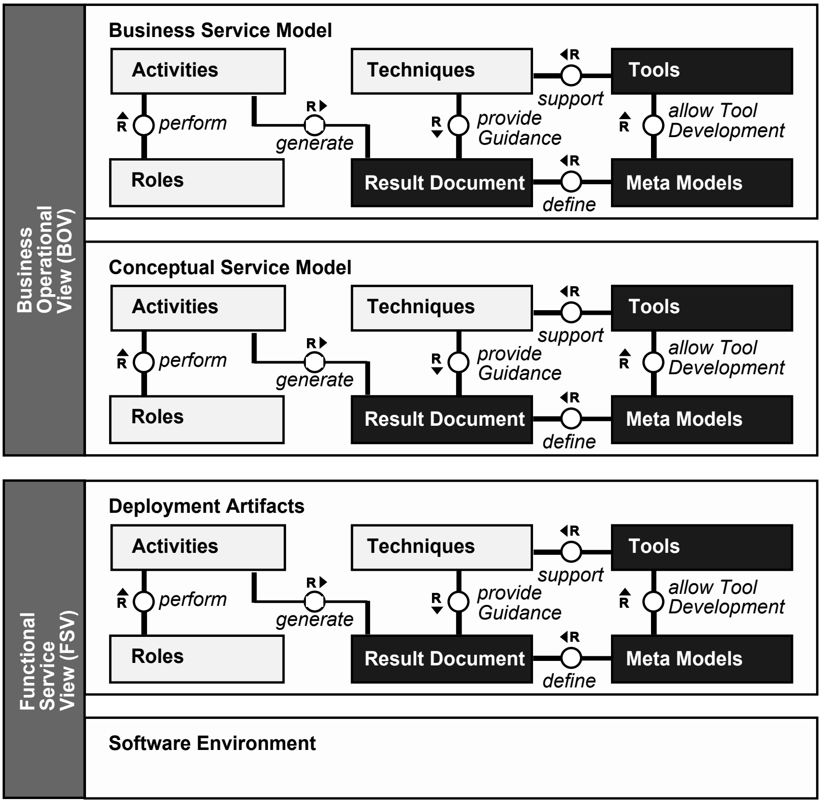


Figure 3.1.: Open-EDI Reference Model & Service Description Layers

final purpose and context. The next layer, the *Conceptual Service Model*, represents the actual modeling purpose of service descriptions. Service properties on this layer reflect a firm establishment with concrete values. The result is a value proposition toward potential customers. *Deployment Artifacts* describe technical specifications to implement service properties. Each layer features the *same* artifacts from method engineering: activities, roles, techniques, result documents, tools, and meta models. Nevertheless, each artifact is implemented differently for each layer.

Table 3.1.: Classification of Service Description Concepts

BSM	**CSM**	**DA**
BMO [98]	Service Property Analysis [79, 100]	WSMO [112]
e^3 value [43]	Service Marketing [22,45,66,71,146]	OWL-S [74]
		WSDL [27]
		SA-WSDL [37]
		UDDI [96]

The method's main objective is the derivation of technical service descriptions from business models. And for doing so, the service description layers offer an appropriate work-break-down structure in order to reduce complexity and to establish a bridge between business and IT. The definition of method engineering artifacts provides a conceptual formalism for service descriptions. Figure 3.1 shows that method engineering artifacts need to be defined for each layer. This is due to the fact that each layer presents a discreet phase in the service description development process. By defining the method engineering artifacts for each layer, it is possible to acknowledge different subject-matter-experts involved in describing services by codifying best-practices, to manage and generate IT specifications, and to offer cohesion between business and IT, which in turn results in less errors, fasten the development process, and makes it comprehensible. This work, however, concentrates on the development of *tools, meta models* as well as *result documents*, and refers to existing work for *activities, techniques* and *roles*.

Table 3.1 shows available concepts for each layer of the reference model, which influences the development of artifacts for the service description method [116, 118]. The following paragraphs briefly describe each layer.

Business Service Model The first layer in figure 3.1 represents the *Business Service Model* as the method's first main phase. Its purpose is to support business strategists while formalizing a novel service idea with a business model. Chapter 4 provides an introduction into business models and introduces a corresponding meta model and modeling notation. The meta model (BSMM), which is applied in section 4.2, holds information about target customers, distribution channels, value objects, and revenue models. The modeling notation (BSMN) in section 4.3 is a graphical notation based on a specific UML Profile [87], which is used to document business service models. Business strategists with the capability to elicit and to judge opportunities in the service market are the main actors for this layer.

Conceptual Service Model The method's second layer is the *Conceptual Service Model* that is neither considered technical nor specific to one or more platforms. This layer's purpose is to transform service ideas into concrete service offers. Business analysts or marketing experts with knowledge about service markets and products take service ideas from the Business Service Model and use them in order to model service offers. Chapter 5 introduces a careful analysis about service properties in business science, information systems as well as computer science, a meta model, and a corresponding modeling notation. The meta model (CSMM), that section 5.2 explains, holds evidence of service functionality, QoS, marketing as well as financial aspects. The then following section 5.3 introduces another modeling notation for this layer that also utilize the UML Profile specification [87].

Deployment Artifact The third layer of the service description method refers to specifications, which are both technical and platform-specific. The *Deployment Artifacts* layer is considered technical in that this layer implements service offerings with deployable technical specifications. Moreover, technical specifications are in general specific to a given platform, such as Java and Web Service technology. The purpose of this layer is to support IT architects with the development of technical service descriptions on the basis of the CSM. Chapter 6 provides an overview of available specifications for describing services in a technical manner [26, 37, 74, 96, 112] and provides abstract mappings between CSM and one technical specification for the automatic generation of this artifact.

Software Environment The *Software Environment* layer, finally, lists available service description runtimes for automatic service deployment and discovery. Possible registries are UDDI [96] or WSMX [111] for semantic web services. This layer, however, is beyond this work.

3.3. Running Example: Eco Calculator

This section presents the *Eco Calculator* service as a running example for the rest of this section. Even though it is an invented service it is a rich scenario, borrowed from the Theseus/TEXO research project [5]. Two scenarios with realistic services are presented in chapter 8.

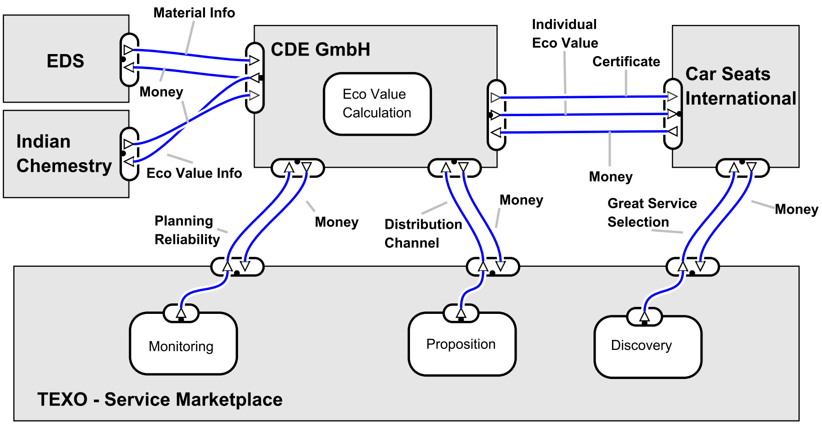


Figure 3.2.: Ecocalculator Scenario

CDE GmbH is a company that offers customized computational services in the ecological domain. Figure 3.2 depicts CDE's business model with the e^3 Value Ontology [44] (cf. section 4.1). *Eco Value Calculation* is its main service. It calculates the carbon dioxide footprint for any given material. In order to offer it, the company relies on two other companies. EDS, for example, provides detailed information about materials and their composition from raw materials. Indian Chemistry, on the other hand, provides a database containing carbon dioxide footprint for raw materials. With this information, CDE is in the position to decompose given material into raw materials, to obtain the footprint for each raw material, and calculate the overall carbon dioxide footprint. Furthermore, in case materials' footprints are in given limits, CDE may issue Eco Certificates to its customers. Car Seats International, who is shown in the figure's upper right corner, is an automotive supplier in Germany that produces and ships car seats for a wide variety of car models. TEXO in the lower part of the figure, is a service marketplace that affords service providers a new distribution channel in that it enlists their services for a fee. Consequently, TEXO provides selected services for a fee to possible service consumers which benefit from this service because it lowers their transaction costs in finding appropriate services, which in this case is Car Seats International. Additionally, the marketplace's third service provides customers with the ability to monitor service execution.

Car Seats International (CSI) targets the Australian automotive market. In order to export car seats to Australia, the company is challenged with a legal restraint that says that any products to be imported into Australia must possess a certificate that states a proper carbon dioxide that stays in certain ranges that are considered safe. Because CSI merely designs and assembles car seats while obtaining their materials from other suppliers, the company has neither information about raw materials' footprint nor final car seats. However, this information is crucial during car seat designs that are intended for the Australian market. Hence, the company looks for a partner who features carbon dioxide footprint calculation for whole car seat designs and is authorized to issue carbon dioxide certificates. In order to find a partner, CSI approaches the TEXO service market place to source for an appropriate service provider that they find in CDE GmbH.

The following chapters utilize this scenario for explaining the service description method from the perspective of CDE GmbH as the service provider.

4. Business Service Model

Whereas the previous chapter provided a complete method overview, this chapter elaborates on the Business Service Model (BSM) with its artifacts. Figure 4.1 depicts a detailing view of the Business Service Model as part of SDM4SE (cf. figure 3.1) as well as the corresponding method engineering artifacts. As aforementioned in section 3.2, the intention of BSM is to grasp service ideas by specifying a business model. In order to complete the first layer, six *activities* need to be performed: (1) establish value offer, (2) constitute value objects, (3) determine target customers, (4) determine relationship for each target customer, (5) determine distribution channel, and (6) setup appropriate revenue models. The two *roles* business strategist and modeling expert perform these six activities in collaboration. Business strategists are subject-matter-experts in a service domain and possess valuable knowledge of service markets, marketing in general, and service trends. Modeling experts, on the other hand, have the ability to elicit and to document the knowledge of business strategists. For doing so, modeling experts rely on a set of *techniques*: service modeling, workshops, and semi-structured interviews. *Tools* such as UML Profiles, UML in general, or spreadsheets support these techniques. The black-shaded method engineering artifacts in figure 4.1 indicate the focus of this chapter. The structure of the remaining chapter is as follows. While section 4.1 analyses available literature in the business model domain, section 4.2 introduces the BSMM, a *meta model* for defining business models with a focus on service descriptions, which is used to grasp services' core ideas. The then following section 4.3 shows how to develop the BSMN, a modeling notation for the meta model based on a UML Profile as a *tool*.

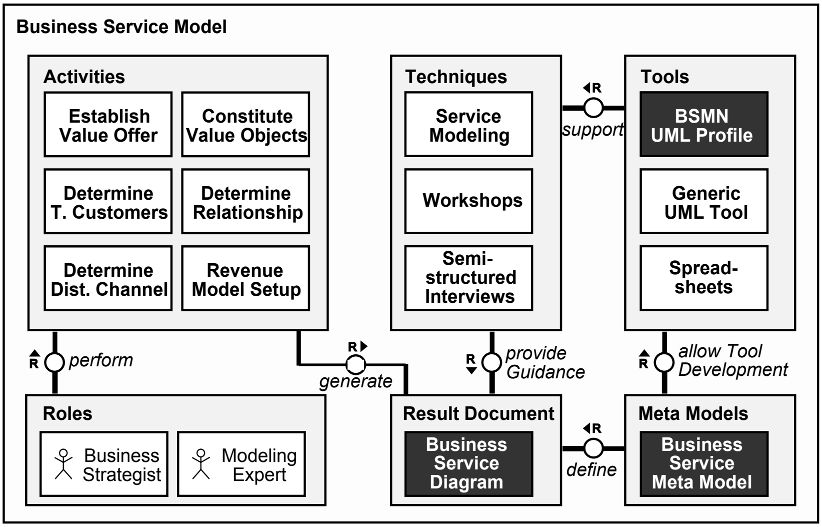


Figure 4.1.: BSM Overview

4.1. Business Model Analysis

This section elaborates on the Business Model Ontology (BMO) [98] from Osterwalder and Gordijn's e^3 Value ontology [44]. Both approaches are suitable for the most abstract layer of SDM4SE. Both ontologies amend each other for Osterwalder rather focuses on single companies, whereas Gordijn concentrates on the co-operation of companies. Evidence for their suitability is found in the literature [33,62,114,116]. The following section uses these ontologies in order to generate the meta model for the BSM layer.

Business Modeling Ontology (BMO) BMO [98] is an ontology to accurately describe companies' business models. Osterwalder did an exhaustive literature analysis of existing business model definitions and theories as well as some real-world case studies in order to build and to evaluate the ontology. The author's main influence is the work from Kaplan and Norton [57] about balanced score cards. The ontology's complexion is that of pillars, building blocks, and attributes. Table 4.1 shows the four pillars in its first row and below the nine corresponding building blocks. The attributes for each building block are not shown in the table but explained below. Osterwalder [98] provides a far more detailed description.

Table 4.1.: Business Model Ontology concepts

Product	**Customer Interface**	**Infrastructure Management**	**Financial Aspects**
Value Proposition	Target Customer	Value Configuration	Cost Structure
	Distribution Channel	Capability	Revenue Model
	Relationship	Partnership	

The main ontology concepts (pillars) include *Product, Customer Interface, Infrastructure Management,* and *Financial Aspects.* The product concept addresses a company's business domain as well as its products and value propositions. The customer interface comprises target customers, means to deliver products to them, and the type of the relationship between companies and their target customers. Network types, infrastructure and logistical performance is represented by the infrastructure management concept. The financial aspect concept includes revenue models and cost structures.

The pillars group nine concepts which Osterwalder refers to as *building blocks.* Like the product concept, the *Value Proposition* refers to companies' products that are valuable to specific customers. Attributes include: (1) name & description, (2) reasoning, (3), value level, (4) price level, and (5) life cycle step. The *target customer* concept supports to differentiate customer segments and select specific segments companies want to address.

Attributes include name & description. In order to specify how to deliver value to customers, the *distribution channel* concept is used. Attributes comprise: (1) reasoning, (2) customer buying cycle, (3) value level, and (4) price level. The *relationship* concept defines the state between companies and their customers, which is formalized with the attribute customer equity. The *value configuration*'s gist is to specify how value for customers is created. Attributes are (1) a configuration type and (2) activities. A *capability* describes companies' knowledge to carry out tasks in order to create customer value. Attributes include (1) name & description and (2) a resource type. Osterwalder described a *partnership* as a cooperative agreement between two or more companies. Attributes include: (1) name & description, (2) reasoning, (3) strategic importance, (4) degree of competition, (5) degree of integration, and (6) substitutability. The *cost structure* specifies monetary expenditure that incur during the value creation process. Its attributes are: (1) name & description, (2) sum, and (3) percentage. The *revenue model* tells how customers compensate received values. Attributes include: (1) name & description, (2) stream type, and (3) pricing method.

e^3 **Value Ontology** Gordijn et al. [44] argue that current requirement engineering methodologies are inadequate for the e-commerce domain, and hence, develop the e^3 Value ontology. e^3 Value ontology offers a structured approach, to gather requirements for e-commerce applications. It includes the ontology itself with its main eight concepts, three levels of abstraction, and a six step process for guidance. An example for applying the ontology is found in section 3.3.

The ontology's main concepts include *actors* which represent other economic entities, *market segments* and *composite actors* that allow the grouping of actors according specific markets and partnerships, respectively. *Value exchanges*, along with *value interfaces* and *value ports*, depict relationships between actors in a business model which offer a means to exchange *value objects* that include, for example, products and money.

The three levels of abstractions are: (1) e-business model development, (2) e-business process design, and (3) software architecture requirements.

Moreover, they provide six steps to guide the requirement creation process: (1) identification of actors in the e-commerce process, (2) construction of the list of the relevant value activities, (3) definition of the associated value ports, interfaces, and value object types, (4) allocation of the value activities to the actors, (5) analysis of the trade-offs occurring in the alternative business models, and (6) tracking down the associated implications for requirements on the information systems architecture.

4.2. Business Service Meta Model

BSMM is a knowledge structure in order to define service descriptions on an abstract level. Scheithauer et al. [117] discuss how this model has been developed using the work of the Business Model Ontology (BMO) [98] as well as the e^3 Value ontology [43], which the previous section introduced. Whereas BMO's focus is on the internal value generation processes, the e^3 Value ontology highlights the value exchange between different actors. The resulting model selects only specific concepts that contribute to a service description, which the next paragraphs introduce: (1) value offer, (2) value object, (3) revenue model (4) distribution channel, and (5) target customer. This meta model is used in order to define result documents as well as to support tool development for BSM in that it specifies what knowledge must be collected during this layer of the method. Figure 4.2 shows the resulting meta model that is explained in the following paragraphs in more detail.

Value Offer is the root element and bundles the following attributes: reasoning, value level, price level as well as life cycle step. `Reasoning` describes in which way a service is valuable for targeted customers. Oster-

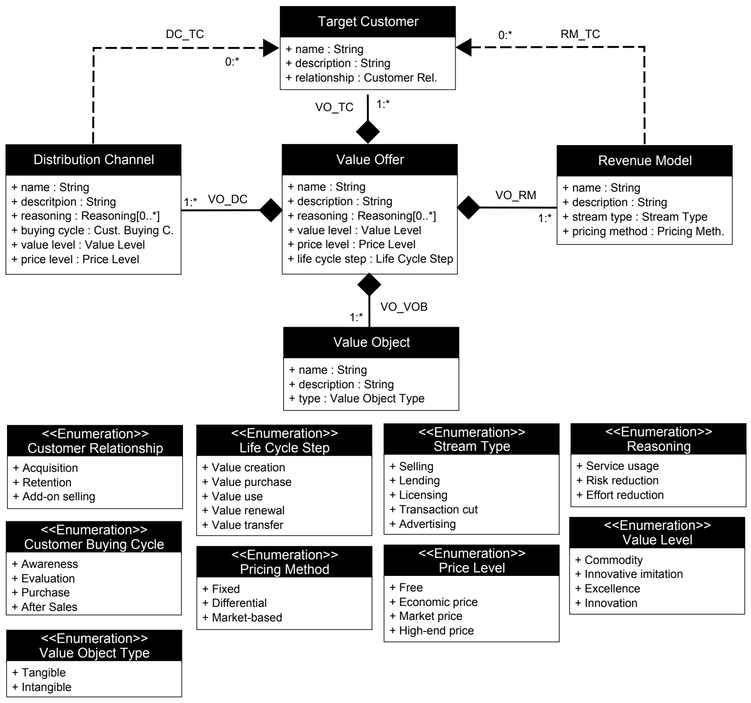


Figure 4.2.: Business Service Meta Model (BSMM)

walder [98] distinguishes three elementary characteristics: value is either created by *using* a service, reducing any kind of *risk* for targeted customers, or reducing customers' *efforts*. The value level states to what extent services distinguish themselves from other companies' offers. Osterwalder provides four possible classifications: either a value offer is a *commodity*, an *innovative imitation*, an *excellence*, or an *innovation*. The price level expresses a services' qualitative pricing strategy. Services are either offered for *free*, for an *economic* (low) price, for an appropriate *market* price, or for a *high-end* price. The life cycle step formalizes when value is created during the service life cycle. Osterwalder explains the life cycle with five steps: *value creation*, *value purchase*, *value use*, *value renewal*, and *value transfer*.

The Eco Value Calculation (EVC) service that section 3.3 introduces, describes a value offer with a *service use* reasoning since the value is created while using the service, and with a *commodity* value level as well as an *economic* price level because it is very easy imitated by competitors.

Value Object is the actual value that is exchanged by companies offering services and companies consuming services. Evidence for this element is found by Osterwalder (called 'Resource') as well as by Gordijn ('Value Object'). In figure 4.2 the aggregation `VO_VOB` specifies that value offers at least offer one or more value objects. Value object's attributes include the value object itself and the value object type. The `type` attribute tells whether the value object is *tangible* or *intangible*.

The EVC service features two value objects. The first one is the intangible value object *Individual Eco Value* that describes the individual calculated carbon dioxide amount for a given material. The second is the tangible *Certificate*, which represents a legal documents that certifies the calculated carbon dioxide.

Revenue Model describes the transformation of value offerings into income. The `VO_RM` aggregation prescribes that a value offer needs to specify at least one or more revenue models. Moreover, the `RM_TC` association tells that revenue models might be linked to the target customer entity so that it is possible to have specific revenue strategies for different target customers. The revenue model entity comprises the following attributes: stream type and pricing method as well as a link to the target customer. The `stream type` attribute formalizes how income is generated. Possible stream types include: *selling, lending, licensing, transaction cut*, and *advertising*. The `pricing method` describes in which way a price is determined. According to Osterwalder, a price is either *fixed* and is agnostic to the environment and customer characteristics, is *differential* and depends on product as well as customer characteristics, or is *market-based* in that the price is determined dynamically between provider and customer.

The EVC service's stream type is *selling* for CDE GmbH sells individual carbon dioxide calculations as well as certificates. Furthermore, the pricing method is set to *fixed* because the price is the same for every potential customer and because it is an *economic* price level without any margin.

Distribution Channel tells how companies deliver value to targeted customers. Value offers must specify one or more distribution channels that the aggregation `VO_DC` shows. Also, distribution channels can link to target customers (cf. association `DC_TC`). The element bundles the attributes: reasoning, value level, price level, and customer buying cycle. The attributes reasoning, value level, and price level have the same semantic as in the value offer bundle, and hence, these can be setup for each channel. The `customer buying cycle` tells which step the channel addresses. Osterwalder proposes four steps for the buying cycle: *awareness*, *evaluation*, *purchase*, and *after sales*.

The *TEXO Service Marketplace* is EVC service's main distribution channel. Its main purpose in the customer buying cycle is the *purchase* step.

Target Customer specifies customer segments. Segments base, for example, on geographical criteria. The `VO_TC` aggregation specifies that value offers need at least one target customer. The `relationship` attribute depicts in detail the type of connection between companies and their target customers. The relationship element classifies target customers according to their equity goals. Osterwalder offers three classes, namely *acquisition*, *retention*, and *add-on selling*. *Producing companies* are CDE GmbH's target customers, such as the Car Seats International company. The relationship to this customer group is set to *acquisition* for there exists no relationship to this customer group, yet.

4.3. Business Service Modeling Notation

Following the BSMM introduction in the previous section, this section elaborates on a corresponding notation. The Business Service Modeling Notation (BSMN) intends to support business strategists and modeling experts while *documenting* and *discussing* business service models, and hence to apply the Business Service Meta Model. Figure 4.7 shows the *BSM result document* for the Eco Calculator example. This section goes through the three step approach, which was introduced in section 2.1.2, in order to develop the notation for BSM.

4.3.1. Step 1: Mapping between BSMM and UML Meta Model

The first step is to establish a mapping between BSMM elements (cf. figure 4.2) and UML meta model elements. This step clarifies how to represent domain-specific elements with UML elements. Three main areas for mapping exist: (1) Classes & Properties, (2) Enumerations & Literals, and (3) Associations. Figure 4.3 exemplifies this. For example, it shows that the element `Target Customer` corresponds to `UML Class`, the element `Customer Relationship` is a `UML Enumeration`, and that `RM_TC` relates to a `UML Association`. Whereas figure 4.3 shows an excerpt of this mapping, tables 4.2 and 4.3 ascribe in detail the mapping between elements of BSMM and the UML meta model. This mapping serves as input for step 2.

4.3.2. Step 2: Meta Model Comparison

With the availability of the mapping between BSMM and the UML meta model from step 1, this step outlines differences between the two meta models. Each discovered discrepancy needs to be considered for the UML Profile generation. The tables 4.2 and 4.3 show deviances for all mappings that the following paragraphs explain in detail.

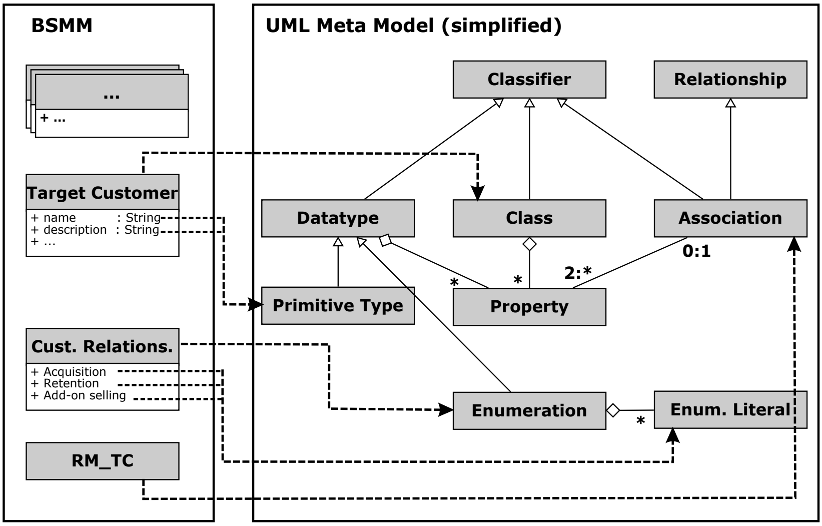


Figure 4.3.: Excerpt of mapping between BSMM and UML Meta Model

Classes & Properties In step 1 identified classes and their properties were mapped to the UML meta model. For example in table 4.3, the `Target Customer` element in line #1 is mapped to UML Class. Since the UML meta model embodies no such element, the Target Customer is marked as *New UML Class*. In contrary, the `Target Customer`'s name property (line #2) was mapped to the existing UML Class property *name*, and hence, equals the UML meta model. However, the `Target Customer`'s property *description* (line #3) may not be directly mapped and is marked with *New Datatype*. The tables 4.2 and 4.3 show that the identification of differences between classes and properties is done in the same manner for all BSMM elements: `Value Offer`, `Distribution Channel`, `Revenue Model`, and `Value Object`.

Enumerations & Literals Likewise, enumerations and their literals are mapped to the UML meta model. E.g., table 4.3 in line #4 shows the

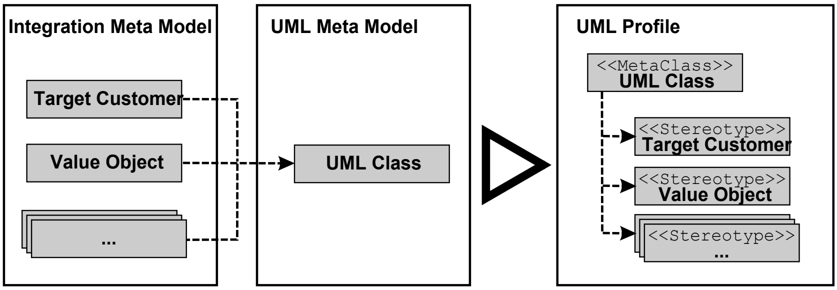


Figure 4.4.: Application of Rule 1

mapping between the BSMM's `Customer Relationship` and the UML `Enumeration` that is marked as a *New Enumeration*. Furthermore, the mappings #5–7 for the literals, *Acquisition*, *Retention*, and *Add-on selling*, are non-existent in the UML meta model, and in consequence, marked as *New Enumeration Literals*. This mapping is similar to the other enumerations, such as, `Customer Buying Cycle`, `Value Object Type`, `Life Cycle Step`, `Pricing Method`, `Stream Type`, `Price Level`, `Reasoning`, and `Value Level`.

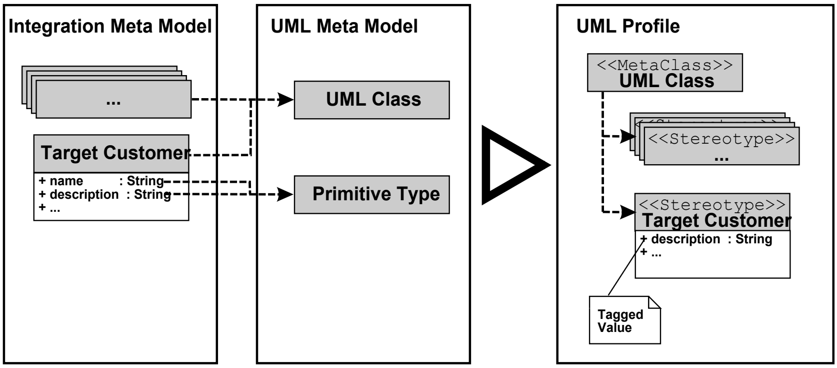


Figure 4.5.: Application of Rule 2

Associations Lastly, associations need to be mapped to the UML meta model. BSMM outlines six relationships for interconnecting classes. For example, the relationship `RM_TC` tells that a `Revenue Model` is valid for none or more `Target Customer` and was mapped with the UML element `Association` (cf. table 4.3 in line #31). However, the difference between BSMM and UML is that in case of the BSMM, the `Revenue Model` may be only associated with `Target Customer`, whereas the UML `Association` defines its `memberEnd.type` with any related element, and thus, the `memberEnd.type` is marked with *memberEnd.type = Target Customer*. This `memberEnd.type` difference is similar for the remaining five relationships `DC_TC, VO_TC, VO_VOB, VO_RM, and VO_DC`.

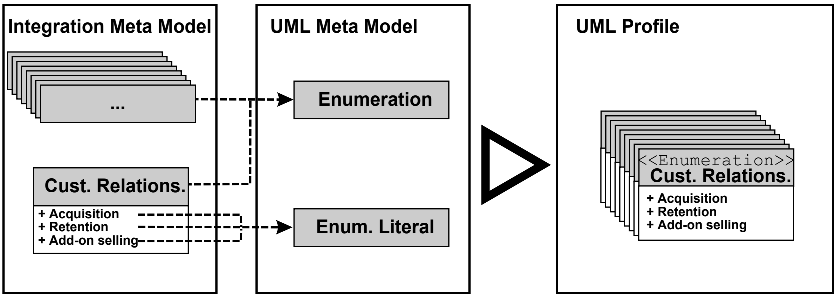


Figure 4.6.: Application of Rule 6

4.3.3. Step 3: Integration Meta Model Transformation

The last step aims at considering the discovered differences in step 2 for the UML Profile with *transformation rules*. Eleven rules (cf. [40]) are the basis. This subsection goes through the rules 1, 2, 6 one by one for classes, properties, and enumerations. Other rules are skipped for they are not necessary for the BSMN.

Rule 1: *(one Stereotype for each equivalent class)* The BSMM shows five main entities. In coherence with rule one, each class is represented with a new `Stereotype`. Figure 4.4 exemplifies that `Target Customer` and `Value Object` are UML Classes and are represented with a Stereotype in the UML Profile definition.

Rule 2: *(one Tagged Value for each new property)* Properties comprise *attributes* as well as *associations*. Tagged values consist of a name and a type. New attributes that were discovered in step 2 will be represented with a tagged value. For example, the class `Target Customer` embodies the new property *description* that is presented as a tagged value: `description: String`. Figure 4.5 shows rule 2's output. It is important to note that the *name* attribute is not represented with a tagged value for this attribute already exists in the `UML Class` element.

Rule 6: *(one Enumeration for each new enumeration with new literals)* Each of the BSMM's enumerations with their literals are acknowledged with a UML Enumeration. Figure 4.6 shows that the element `Customer Relationship` is an Enumeration, and that its attributes `Acquisition`, `Retention`, and `Add-on selling` are Enumeration Literals.

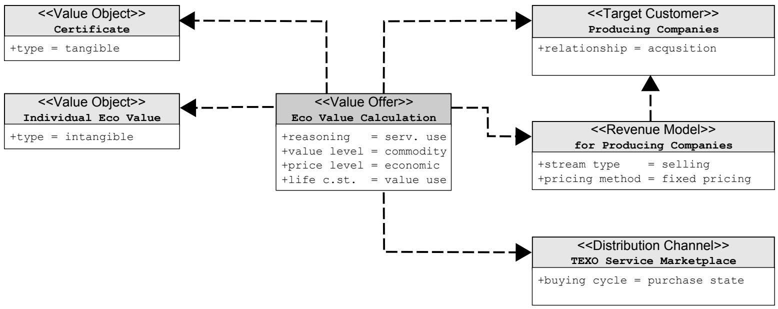


Figure 4.7.: Business Service Diagram for Eco Value Calculation Service

4.3.4. BSM Result Diagram

It is possible to use the generated UML Profile BSMN with every UML-compliant UML Tool, including the Eclipse UML2 Toolset [1]. Chapter 7 discusses the Business Service Modeling Tool that implements this modeling notation. Figure 4.7 shows the business service diagram for the Eco Value Calculation Service that was modeled with the BSMN UML Profile. It shows the value offer *Eco Value Calculation* with its characteristics in the middle of the figure. On the left are the two value offers: the tangible *Certificate* and the intangible *Individual Eco Value*. In the figure's top right corner is the target customer *Producing Companies* with its acquisition relationship. Below is the corresponding revenue model *for Producing Companies* that indicates a *selling* stream type as well as a *fixed pricing* method. The figure's lower right corner displays the *TEXO Service Marketplace* distribution channel that supports customers' *purchase state*.

Table 4.2.: BSMM to UML Mapping and Differences for Value Offer.

#	BSMM	MAPPING	DIFFERENCES
1	**Value Offer**	UML Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	**reasoning**	Enumeration	New UML Enumeration
5	Service usage	Enumeration Literal	New Enumeration Literal
6	Risk reduction	Enumeration Literal	New Enumeration Literal
7	Effort reduction	Enumeration Literal	New Enumeration Literal
8	**value level**	Enumeration	New UML Enumeration
9	Commodity	Enumeration Literal	New Enumeration Literal
10	Inno. imitation	Enumeration Literal	New Enumeration Literal
11	Excellence	Enumeration Literal	New Enumeration Literal
12	Innovation	Enumeration Literal	New Enumeration Literal
13	**price level**	Enumeration	New UML Enumeration
14	Free	Enumeration Literal	New Enumeration Literal
15	Economic price	Enumeration Literal	New Enumeration Literal
16	Market price	Enumeration Literal	New Enumeration Literal
17	High-end price	Enumeration Literal	New Enumeration Literal
18	**life cycle step**	Enumeration	New UML Enumeration
19	Value creation	Enumeration Literal	New Enumeration Literal
20	Value purchase	Enumeration Literal	New Enumeration Literal
21	Value use	Enumeration Literal	New Enumeration Literal
22	Value renewal	Enumeration Literal	New Enumeration Literal
23	Value transfer	Enumeration Literal	New Enumeration Literal
24	**VO_TC**	Association	New Property
25			memberEnd.type = Target Customer
26			memberEnd.lower = 1
27	**VO_RM**	Association	New Property
28			memberEnd.type = Revenue Model
29			memberEnd.lower = 1
30	**VO_VOB**	Association	New Property
31			memberEnd.type = Value Object
32			memberEnd.lower = 1
33	**VO_DC**	Association	New Property
34			memberEnd.type = Distribution Channel
35			memberEnd.lower = 1

Table 4.3.: BSMM to UML Mapping and Differences for Target Customer, Distribution Channel, Revenue Model, and Value Object.

#	BSMM	MAPPING	DIFFERENCES
1	**Target Customer**	UML Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	**relationship**	Enumeration	New UML Enumeration
5	Acquisition	Enumeration Literal	New Enumeration Literal
6	Retention	Enumeration Literal	New Enumeration Literal
7	Add-on selling	Enumeration Literal	New Enumeration Literal
8	**Distribution Channel**	UML Class	New UML Class
9	name	Primitive Datatype	N/A
10	description	Primitive Datatype	New Datatype
11	**buying cycle**	Enumeration	New UML Enumeration
12	Awareness	Enumeration Literal	New Enumeration Literal
13	Evaluation	Enumeration Literal	New Enumeration Literal
14	Purchase	Enumeration Literal	New Enumeration Literal
15	After Sales	Enumeration Literal	New Enumeration Literal
16	**DC_TC**	Association	New Property
17			memberEnd.type = Target Customer
18	**Revenue Model**	UML Class	New UML Class
19	name	Primitive Datatype	N/A
20	description	Primitive Datatype	New Datatype
21	**steam type**	Enumeration	New UML Enumeration
22	Selling	Enumeration Literal	New Enumeration Literal
23	Lending	Enumeration Literal	New Enumeration Literal
24	Licensing	Enumeration Literal	New Enumeration Literal
25	Transaction cut	Enumeration Literal	New Enumeration Literal
26	Advertising	Enumeration Literal	New Enumeration Literal
27	**pricing method**	Enumeration	New UML Enumeration
28	Awareness	Enumeration Literal	New Enumeration Literal
29	Evaluation	Enumeration Literal	New Enumeration Literal
30	Purchase	Enumeration Literal	New Enumeration Literal
31	**RM_TC**	Association	New Property
32			memberEnd.type = Target Customer
33	**Value Object**	UML Class	New UML Class
34	name	Primitive Datatype	UML Class :: name (No Difference)
35	description	Primitive Datatype	New Datatype
36	**type**	Enumeration	New UML Enumeration
37	Tangible	Enumeration Literal	New Enumeration Literal
38	Intangible	Enumeration Literal	New Enumeration Literal

5. Conceptual Service Model

While the previous chapter introduced the BSM as the most abstract layer of the service description method, this chapter outlines the Conceptual Service Model (CSM). Figure 5.1 depicts a detailing view of the Conceptual Service Model as part of SDM4SE (cf. figure 3.1) as well as the corresponding method engineering artifacts. The intention of CSM is to transform service ideas and business models into concrete service descriptions which are neither technical nor platform-specific.

As shown in figure 5.1, CSM ascribes eight abstract *activities* in order to complete this layer: (1) establish service product, (2) define process properties, (3) provide people information, (4) determine physical evidence, (5) setup channels, (6) establish pricing, (7) establish promotion properties, and (8) adjust productivity and quality. CSM furthermore specifies two different *roles* who are responsible during these activities. Marketing experts are subject-matter-experts in defining a marketing proposal for promoting services on market places with sophisticate knowledge of promotion, pricing, as well as service trends on the basis of business models. Modeling experts own the ability to elicit and to document this knowledge of marketing experts. For doing so, modeling experts rely on a set of *techniques*: service modeling, workshops, and usage of the 8 Ps in service marketing [71]. *Tools* such as UML Profiles, UML in general, or spreadsheets support these techniques.

The black-shaded method engineering artifacts indicate the focus of this chapter. The rest of this chapter is structured as follows. While section 5.1 analyses available literature in the service marketing domain [71] as well as in information systems and computer science, section 5.2 in-

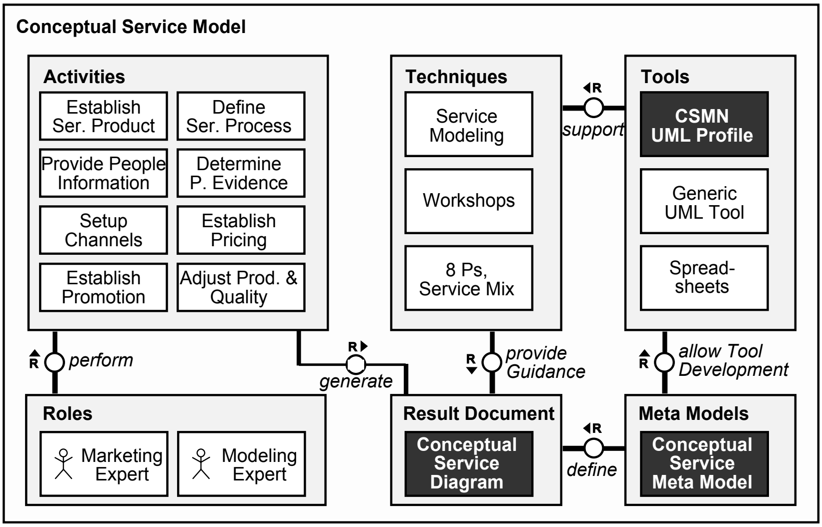


Figure 5.1.: CSM Overview

troduces the Conceptual Service Meta Model, a consolidated result of the literature analysis in form of a *meta model* for defining conceptual service descriptions, which is used to define service offerings. The then following section 5.3 shows how to develop the Conceptual Service Modeling Notation, a modeling notation for the meta model based on a UML Profile as a *tool*.

5.1. Property Analysis

This section introduces the service property model [116], which is appropriate for the CSM layer. Scheithauer et al. [116] investigate valid service properties for service ecosystem, available modeling notations for service properties, and an appropriate framework in order to categorize modeling notations according to the needs of the different roles involved in the service description development process. The motivation to find valid service properties is to describe services in a way suitable for aimless service trade over the Internet. It is envisioned that these properties support service proposition, discovery, selection, contracting, and monitoring in service ecosystems. In order to elicit valid service properties, this section investigates available literature and analyzes proposed properties in three disciplines. The result are 39 service properties and their relationships. For a better understanding and readability these properties are grouped into the Eight Ps from the service marketing mix: (1) product, (2) process, (3) people, (4) physical evidence, (5) place and time, (6) price, (7) promotion, and (8) productivity and quality.

The following paragraphs introduce the analysis framework, the analyzed disciplines, as well as the service mix for property categorization and the data collection method. This analysis is the basis for the Conceptual Service Meta Model that the next section presents.

Analysis Framework Two dimensions build the analysis framework: *disciplines* and *service mix*. These dimensions allow a further reasoning about service properties and their importance in a research discipline as well as to provide a context for property interpretation (cf. tables 5.1–5.3). For example, service quality has a different meaning in business science than in computer science.

Different Disciplines Considered are three disciplines in order to analyze literature about service properties. Authors who wrote about service marketing and/or belong to a business science department categorize for *Business Science (BS)*. Consequently, authors who are from computer science departments and/or wrote about network-specific service properties belong to *Computer Science (CS)*. Lastly, authors who used models or ontologies to formalize business knowledge for testing, business-IT alignment, or similar categorize for *Information Systems (IS)*. The first two rows of tables 5.1–5.3 depict the three disciplines with their 26 publications.

Service Mix – The Eight Ps is the other dimension that has its root in service marketing research for the last 15 years. According to Lovelock and Wright [71], the service mix with its eight components captures the notion of services. Here, the service mix performs a simple grouping of identified service properties as shown in the first two columns of tables 5.1–5.3.

- *Product* embodies services' core information including information about supplementary services and benefits.
- *Process* describes how products are delivered to customers. They embody activities and their sequence as well as how customers interact with services.
- Service delivery depends on *People* such as sales people. Naturally, customers perceive service quality in terms of service personnel. Furthermore, other actors such as partners categorize as people as well.
- Next to services themselves, *Physical Evidence* plays an important role in service marketing. This component addresses tangibles with an impact on customers, such as buildings, landscapes, vehicles, signs, and others.
- Service delivery involves information about *Place and Time* as well as delivery and supply channels.
- *Price* embodies decisions about pricing methods, demand and supply, differentiation, and ways of payment.
- *Promotion* has two important parts: information and advice, to persuade target customers, and to encourage customers to perform buying actions. This includes mainly sales people and advertising.

- *Productivity and Quality* tells customers how well a service performs and is measured in customer added value as well as to which level the customer is satisfied with services' outputs.

Data Collection Method The analyzed data were collected during a two years research project about service engineering. At first, the collection method was broad in the sense that different publication repositories were consulted with the following keywords: *service properties, service attributes, Quality of Service,* and *service description.* Only publications that had information about properties that describe services were kept. Following this, the collection method became investigative in the way that references from the remaining publications were evaluated for novel properties. Finally, a set of 26 publications remained from originally over 85 publications.

Tables 5.1, 5.2, and 5.3 show the resulting service properties for each publication. A final discussion about each property for each author is beyond this work. However, the next section presents a final understanding of these properties in one conceptual model.

Table 5.1.: Property Analysis in Business Science, Information Systems, and Computer Science (Part 1/3)

		BS					IS							CS															
		Mörschel & Höck 2002 [76]	Lovelock & Wright 2002 [71]	Grönroos, 2007 [45]	Kotler & Keller 2007 [66]		Dumas et al. 2001 [35]	Baida et al. 2003 [14]	Oaks et al. 2003 [79]	O'Sullivan et al. 2005 [100]	de Miranda et al. 2006 [30]	Weigand et al. 2009 [140]		Chung et al. 1994 [28]	Barbacci et al. 1995 [17]	DCMI, 1998 [6]	IEEE 830-1998 [127]	Keller & Ludwig 2003 [59]	Lee et al. 2003 [70]	Avizienis et al. 2004 [13]	UDDI, version 3.0.2 2004 [96]	Zeng et al. 2004 [147]	Papaioannou et al. 2006 [104]	Brien et al. 2007 [95]	Giallonardo & Zimeo 2007 [41]	Momotko et al. 2007 [75]	QWS Dataset 2007 [7]	WSDL 2.0 2007 [27]	UPMQoS 2008 [92]
PRODUCT	ID	x								x						x					x								
	Name / Title								x	x						x					x						x	x	
	Language			x						x						x													
	Documentation			x												x					x						x		
	Version															x													
	Dates															x													
	Type															x													
	Classification	x	x	x	x		x	x	x			x									x						x		
	Terms of Use	x														x													
	Benefit		x	x																									
	Suppl. S.		x																										
	Customization		x																					x					
PROCESS	Capability	x							x														x		x			x	
	Interface	x																											
	Duration	x																	x			x							
	Input / Output		x					x	x																			x	
	Supp. Stand.									x													x						

Table 5.2.: Property Analysis in Business Science, Information Systems, and Computer Science (Part 2/3)

		PEOPLE						**P. EV.**				**P.&T.**			**PRI.**		
		Rights	Provider	Consumer	Address / E. / P.	Creator	Signature	Resource	Condition	Resource Type	Reports	Channel	Temporal Entity	Location Entity	Price / Pricing	Pricing Methods	Payment
BS	Mörschel & Höck 2002 [76]			x				x			x			x	x		
	Lovelock & Wright 2002 [71]			x				x				x			x		x
	Grönroos, 2007 [45]			x				x							x		
	Kotler & Keller 2007 [66]			x								x			x		
IS	Dumas et al. 2001 [35]		x									x			x		x
	Baida et al. 2003 [14]							x		x					x		x
	Oaks et al. 2003 [79]								x					x			
	O'Sullivan et al. 2005 [100]	x	x										x	x	x	x	x
	de Miranda et al. 2006 [30]														x	x	
	Weigand et al. 2009 [140]																
CS	Chung et al. 1994 [28]																
	Barbacci et al. 1995 [17]																
	DCMI, 1998 [6]	x				x											
	IEEE 830-1998 [127]																
	Keller & Ludwig 2003 [59]																
	Lee et al. 2003 [70]																
	Avizienis et al. 2004 [13]																
	UDDI, version 3.0.2 2004 [96]				x		x										
	Zeng et al. 2004 [147]														x		
	Papaioannou et al. 2006 [104]																
	Brien et al. 2007 [95]																
	Giallonardo & Zimeo 2007 [41]																
	Momotko et al. 2007 [75]		x												x		x
	QWS Dataset 2007 [7]																
	WSDL 2.0 2007 [27]																
	UPMQoS 2008 [92]																

Table 5.3.: Property Analysis in Business Science, Information Systems, and Computer Science (Part 3/3)

		BS					IS							CS															
		Mörschel & Höck 2002 [76]	Lovelock & Wright 2002 [71]	Grönroos, 2007 [45]	Kotler & Keller 2007 [66]		Dumas et al. 2001 [35]	Baida et al. 2003 [14]	Oaks et al. 2003 [79]	O'Sullivan et al. 2005 [100]	de Miranda et al. 2006 [30]	Weigand et al. 2009 [140]		Chung et al. 1994 [28]	Barbacci et al. 1995 [17]	DCMI, 1998 [6]	IEEE 830-1998 [127]	Keller & Ludwig 2003 [59]	Lee et al. 2003 [70]	Avizienis et al. 2004 [13]	UDDI, version 3.0.2 2004 [96]	Zeng et al. 2004 [147]	Papaioannou et al. 2006 [104]	Brien et al. 2007 [95]	Giallonardo & Zimeo 2007 [41]	Momotko et al. 2007 [75]	QWS Dataset 2007 [7]	WSDL 2.0 2007 [27]	UPMQoS 2008 [92]
PROM.	Discount		x							x																			
	Test Report																												
	Rating / Trust			x						x												x							
	Certificate									x																			
PRODUCTIVITY & QUAL.	Dependability			x											x					x									x
	Availability						x			x					x		x	x	x	x		x	x	x	x		x		x
	Reliability		x	x											x		x		x	x		x	x	x	x	x	x		x
	Maintainability													x			x			x									
	Accuracy													x					x										
	Performance			x										x	x		x		x				x	x	x				x
	Capacity														x				x										
	Throughput														x			x	x				x				x		x
	Latency														x				x				x				x		x
	Security			x			x			x				x	x		x		x					x	x				x
	Authentication																		x										
	Authorization																		x	x									
	Confidentially			x						x					x				x	x									x
	Data Integrity														x				x	x					x				x

5.2. Conceptual Service Meta Model

This section presents the Conceptual Service Meta Model (CSMM). This meta model formalizes the identified service properties throughout the three disciplines: business science, information systems, and computer science. CSMM serves as a formal grounding for describing business services in order to propose and to discover them in a technology-free manner. Whereas the CSMM ought to abstract from technical specifications including WSDL and UDDI, the intention is to be able to generate these technical specifications from CSMM instances. It is important to note that these properties do not intend to describe services' behavior, implementation, nor how to technically integrate a service into various software environments. They rather serve to propose a service on a market place toward potential customers, to locate and analyze a service, and to compare different services [35]. The aforementioned service mix with its eight categories serves as a classification system that reduces the overall complexity of all these elements. Table 5.4 shows the eight categories along with the corresponding service properties and enumerations (emphasized) that the next subsections explain.

5.2.1. Product

The product category embodies services' core information including information about supplementary services, classifications, terms of use, and benefits. Figure 5.2 shows the corresponding part of the meta model.

Service Product The *service product* property is the root element of each service description. It embodies core attributes with a functional character. Table 5.5 provides an overview of the service product's attributes. Services have a qualified `name` and a `description`. CDE's service, for example, is named *Eco Value Calculation (EVC)* and further described as *calculates the carbon dioxide footprint for any given material.* For identifi-

Table 5.4.: Conceptual Service Meta Model Entities & Enumerations

PRODUCT	**PROCESS**	**PEOPLE**	**P. EVI.**
Service Product	Capability	Provider	Resource
Classification	Standard	Consumer	Condition
Terms of Use	*Interface Type*	Partner	*Resource Type*
Benefit		Right	
Currency		Contact & Addr.	
Service Type		*Right Type*	
Automation Level			
Classification System			
Lang (ISO 639-2)			

PLACE & TIME	**PRICE**	**PROMOTION**	**PROD. & Q.**
Channel	Price	Test Report	Quality
Recurrence	Transaction Cut	Rating	Dependability
Channel Type	Pricing	Certificate	Performance
Time Granularity	Usage-based Pr.	Discount	Throughput
	Flatrate	Payment Disc.	Latency
	Two-Part Tariff	Seasonal Disc.	Security
	N-Block-Tariff	Channel Disc.	Confidentially
	Payment	Quantitative Disc.	Data Integrity
	Payment Instr.	*SERVQUAL*	
	Price Modifier	*8Ps*	
	Currency		

cation, services own a unique `key` attribute that conforms to UDDI [96] or DCMI [6]. The EVC service key is *SVC_KEY_ECOCALC_7493*. The service product also specifies languages that are supported by services. The *Lang* enumeration refers to the ISO 639-2 specification which regulates the representation of languages. For example *ger* for German or *eng* for English. Whereas `written languages` tells in which languages service documents are available, `spoken languages` tells potential service consumers in what language humans can interact with services. Both attributes conform with the ISO 639-2 specification. The EVC service offers merely *english* as spoken and written language. The attributes `version`, `created`, `updated`, and `next update` situate the current status and maturity of services. The EVC service comes in version *2*, was created on *2010-03-09*, last updated on *2010-03-09*, and will receive the next update at

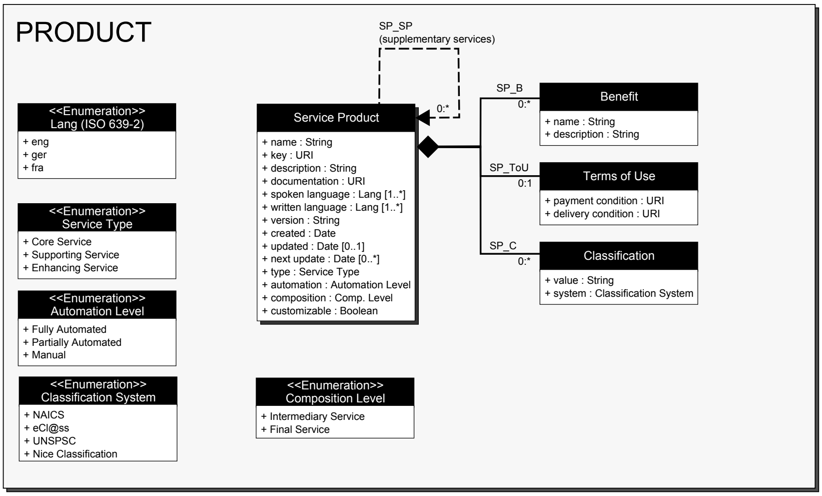


Figure 5.2.: Conceptual Service Meta Model – Product

2010-11-01.

The `Service Type` further characterizes services. According to Baida et al. [14], a *core service* represents a service that adds value to customers' value chain. *Supporting services*, on the other hand, are services that enable customers to access core services in the first place. *Enhancing services*, finally, differentiate core services from competitors offers in that they improve overall services' value. From the perspective of the CDE GmbH, the EVC service is rather a *core service*. The `SP_SP` relationship allows to connect a service product with other service products with the service types *enhancing service* and *supporting service*. In the example scenario, the EDS and Indian Chemistry companies would offer supporting services for CDE GmbH.

According to Dumas et al. [35], services can be categorized according to their `automation level`. *Fully automated* services involve no human interaction. These include transaction services and persistence services. *Partially automated* services, on the other hand, are only automated to a specific portion and involve human interactions. For example, e-commerce retailing is a partially automated service. Finally, *manual services* are fully performed by humans, such as repair services or hair-dresser. The EVC service is a *fully automated* service because all steps involve no humans. Dumas et al. [35] also specify services according to their `level of composition`. They explain *intermediary* services as services with the intention of being part of a composition and not intended for end-consumers, such as equipment repair or persistency services. *Final* services, however, are directly consumable by end-consumers, including for example banking services.

Next to the aforementioned `SP_SP` relationship, service products comprise three other relationships to product properties. It is possible to define none or more benefits with the `SP_B` relationship. The `SP_ToU` relationship delineates the optional terms of use property. Finally, according to the `SP_C` relationship a service product may feature none or more classification properties.

Classification Both disciplines, business science and information systems recommend the use of service *classification systems*, most notably the work of Hepp et al. [48]. From a service provider's view, classification systems allow an appropriate placement of services. Potential consumers use classifications to narrow down the number of available services. All in CSMM supported classification systems make use of industry sectors. As shown in figure 5.2 the classification property features two attributes. The `value` attribute is a placeholder for a unique identifier from the selected `classification system`. CSMM supports four classification systems: (1) North American Industry Classification System (NAICS), (2) eCl@ss, (3) United Nations Standard Products and Services Code® (UNSPSC), and (4) Nice Classification. The EVC service, for example, uses the UNSPSC classification. The value is *12142104* that stands for *Carbon dioxide gas CO2*.

Table 5.5.: Service Product Attributes

Attribute	Type	Description
`name`	`String`	Allows naming services.
`key`	`URI`	Represents a unique ID for services.
`description`	`String`	Allows a further and more detailed textual description of services.
`documentation`	`URI`	A reference to further available information about services.
`spoken language`	`Lang`	Specifies which spoken languages are supported by services' personnel.
`written language`	`Lang`	Specifies which written languages are supported in terms of services' documentation.
`version`	`String`	Documents services' current version.
`created`	`Date`	Documents services' creation date.
`updated`	`Date`	Documents services' update dates.
`next update`	`Date`	Documents the next available update for a service.
`type`	`Service Type`	Specifies whether a service is a *core service*, a *supporting service*, or an *enhancing service*.
`automation`	`Automation Level`	Specifies whether a service is a *fully automated service*, a *partially automated service*, or a *manual service*.
`composition`	`Comp Level`	Specifies whether a service is an *intermediary service*, or a *final service*.
`customizable`	`Boolean`	Tells whether providers allow consumers to customize services.

Terms of Use *Terms of use* is a concept that is found in the work of Mörschel and Höck [76] and O'Sullivan [99] in terms of rights and obligations. Terms of use, however, highly depends on local legal regulations and is impossible to generalize. A step in this direction is, for example, the work of Hoekstra et al. [50]. The CSMM limits itself to references for `payment conditions` and `delivery conditions`. The EVC service points to *http://62.52.175.245:8080/texo/ services/ecocalc/paymentcondition* and *http ://62.52.175.245:8080/texo/ services/ecocalc/deliverycondition*, respectively.

Benefit The *benefit* attribute describes value that goes beyond tangible service outputs. Weigand et al. [140] refer to them as second-order values. A typical example for such a benefit discloses figure 3.2: The value exchange *planning reliability* between CDE GmbH and the TEXO service marketplace. Whereas Grönroos [45] refers to benefits as a part of customer communication, Lovelock and Wright [71] understand them as parts of the service product. CSMM allows providers to `name` and to `describe` such benefits.

5.2.2. Process

Service processes generally describe services in terms of behavior. They embody activities and their sequence as well as how customers interact with services. Grönroos [45], for example, defines a service as a process: *"A service is a process consisting of a series of more or less intangible activities that normally, but not necessarily always, take place in interactions between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems".* Defining service processes in order to deliver value is out of scope of CSMM. It rather focuses on service capabilities and supported standards. Nevertheless, it is the intention to link service processes to service capabilities for a complete conceptual view on services. Scheithauer et al. [117] describe this link as part of a holistic service engineering approach. Figure 5.3 shows the corresponding meta model section.

Capability *Capabilities* describe the different ways of creating value for potential consumers (cf. [76]). For CSMM, a capability represents partially or completely a service's functionality. A service product has one or more capabilities. A capability allows a service consumer to access services' functionality. Often, services' functionality is divided into several capabilities. This allows service consumers to access particular subsets of services' functionality. Additionally, a service's outcome might be different, depending which capabilities are performed in what order. In some cases just some of these capabilities are necessary for service consumers

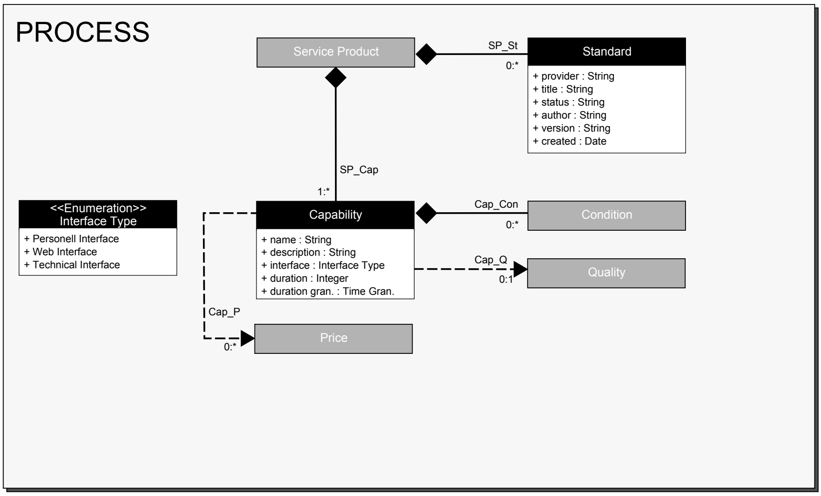


Figure 5.3.: Conceptual Service Meta Model – Process

to achieve their goals.

The `SP_Cap` relationship ascribes that at least one capability needs to be defined for each service product. Next to `name` and to `describe` capabilities, the `interface type` tells how to interact with a service. CSMM differentiates between three interface types. *Personnel interface* refers that a capability can be accessed by approaching a human (e.g. bank teller). *Web interface* describes any website that allows access to services' functionality (e.g. Amazon.com). *Technical interface*, finally, refers to other technical means in order to interact with services. This includes web services, telephone, or fax, just to name a few. Furthermore, capabilities and their corresponding processes own a `duration` attribute [76, 147] that is measured with a `time granularity`.

Additionally, capabilities own pre- and postconditions (`Cap_Con` relationship), and refer to the price (`Cap_P` relationship) and the quality property which the following sections explain.

The EVC service features two capabilities with a *technical interface* for they are web services. The first one is *Calculate Carbon Dioxide* with a duration of *one minute*. The second is *Issue Carbon Dioxide Certificate* with a duration of *one hour*.

Standard According to O'Sullivan [99] and Papaioannou et al. [104], *standards* refer to processes that conform to rules prescribed by a standardization body. Following standards influences the trust relationship between providers and potential consumers. CSMM allows specifying standards with information concerning the `provider`, `standard title`, `status`, `authors`, `version`, and `creation date`. Each service product defines either none or more standards through the `SP_St` relationship.

The EVC service, for example, features the *ISO 9001* standard, which states that the CDE GmbH developed a quality management system.

5.2.3. People

This category embodies properties which deal with the different stakeholders in service marketing as well as legal rights. Stakeholders, or actors, include provider, consumer, and partner. Figure 5.4 shows the corresponding section of the meta model.

Actor *Actor* is a general property for stakeholders. Specializations are the provider property, the consumer property, and the partner property. Often it is not possible to use such a specialization for many actors provide *and* consume services. However, for service descriptions that is not the case. Service descriptions are developed by a certain actor who is then in the role of the provider. By using the `SP_A` relationship it is possible to define none or more actors for each service product.

All attributes strongly relate to UDDI [96]. It is possible to `name` and to `describe` actors. For identification, actors own a unique `key` attribute that conforms to UDDI. Another form of identification is a `DUNS` number [100]. The private US company Dun & Bradstreet provides the Data Univer-

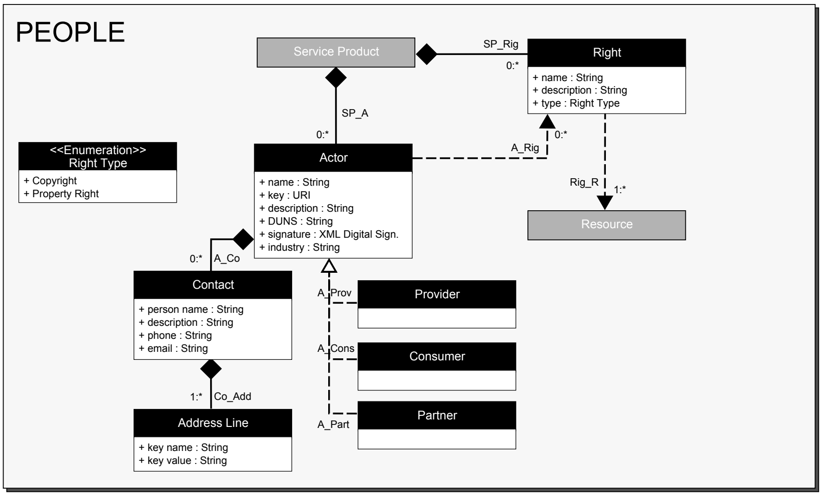


Figure 5.4.: Conceptual Service Meta Model – People

sal Numbering System (DUNS) that manages unique numeric identifiers for business actors in order to assign additional information to companies, e.g., credit information. UDDI also acknowledges the importance of authentication in the business context. For doing so it mandates the `signature` attribute along with a W3C recommendation that specifies signatures on the basis of XML. The `industry` attribute, finally, is motivated by Kotler and Keller [66] and allows a further classification of actors according to their *branch of trade*. Furthermore, the `A_Co` relationship lets modeling experts define further contact information for each actor.

The **Provider** [6, 35, 96, 100] specializes actors into the role of providing services. The **Consumer** [66, 71, 76] are potential customers of services. Service provider may name additional actors they **Partner** [6] with. Partners either support or enhance a provider's offering (cf. service product in section 5.2.1).

Table 5.6 shows the actors for the EVC example service. Information that is not contained in the scenario description are marked with `N/A`.

Table 5.6.: Actors for Eco Value Calculation Service

	Partner	Partner	Provider
name	EDS	Indian Chemistry	CDE GmbH
key	ACT_KEY_1	ACT_KEY_2	ACT_KEY_3
DUNS	N/A	N/A	N/A
signature	N/A	N/A	N/A
industry (NAICS)	Other Information Services	Natural Gas Distribution	Carbon dioxide manufacturing
person name	N/A	Radhika Srinagar	Christian Schubert
phone	N/A	N/A	N/A
email	N/A	N/A	N/A
Add. Line: Street	Mailbox 1234	19, Mahatma Ghandi Avenue	Heinrich-Hertz Str. 19
Add. Line: ZIP	65402	400093	75015
Add. Line: City	Ruesselsheim	Mumbai	Bretten
Add. Line: Country	Germany	India	Germany

Contact and Address UDDI [96] provides means to describe *contact* and *address* information. Actors can hold further contact information. **Contacts** can be `named` and `described`, and own a `phone number` and `email details`. In line with UDDI, **Address Lines** with the `Co_Add` relationship are specified with a `key-value` pair to express details including street, ZIP codes, and cities.

Right Actors can hold rights to *Resources* that the next subsections explain in detail. The `A_Rig` relationship specifies this. However, resources are either tangible or intangible values which are exchanged between actors [44, 98]. The Right property specifies who *owns* these resources. Rights can be `named` and `described` with text. DCMI [6] distinguishes between two different `types`: Copyrights and Property Rights. The right property is connected to at least one or more resources with the `Rig_R` relationship.

5.2.4. Physical Evidence

Grönroos [45], Lovelock and Wright [71], Mörschel and Höck [76] as well as Baida et al. [14] identify the *physical evidence* concept next to the intangible services process as the fourth service marketing category. Figure 5.5 shows that this category embodies two service properties: Resources and Conditions.

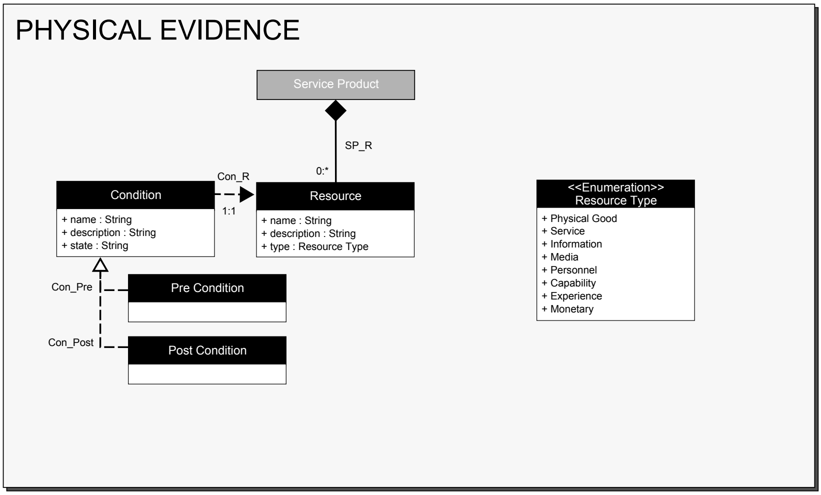


Figure 5.5.: Conceptual Service Meta Model – Physical Evidence

Resource Mörschel and Höck [76], for example, describe resources as *production factors*, which are employed during service execution. These factors include material, services, information, media, and personnel. Lovelock and Wright [71], on the other hand, understand resources as *tangibles* or *visuals* that represent semi-results or results of a service interaction for consumers. Grönroos [45] interprets resources as *tangibles* as in physical facilities, personnel appearance, tools, physical representation, or other customers. Baida et al. [14], finally, infer resources as what services offer and distinguish between different types of resources. With CSMM, Resources are `named` and `described` with text. Additionally, it is possible to `type` resources as physical goods, other services, information, media, personnel, capability, experience, and monetary. For each service product it is possible to define none or more resources with the `SP_R` relationship as shown in figure 5.5.

Three resources exist in the EVC service example. The information resource *Individual Eco Value* and the physical-good resource *Material* are used in the *Calculate Carbon Dioxide* capability. The *Certificate* is a physical-good resource that is employed in the *Issue Carbon Dioxide Certificate* capability.

Condition While resources describe entities in general that are utilized by services, it is often the case that they are used differently by services' capabilities. Mörschel and Höck [76] differentiate between an as-is-state and a target-state of resources. Oaks et al. [79], on the other hand, ascribe pre- and postconditions for resources, acknowledging that services processes may alter resources during service execution. Hence, CSMM's *condition* property describes how resources are utilized by capabilities. Each condition relates to exactly one resource that is specified with the `Con_R` relationship. Conditions are `named` and `described` with text. Furthermore, the `state` attribute holds information about the as-is-state or the target-state. The condition entity is an abstract property that generalizes the Pre-Condition and the Post-Condition properties. Whereas pre-conditions specify resources' as-is-state for capabilities, post-conditions specify resources' target-state.

The EVC example service offers two capabilities. The capability *Calcu-*

late Carbon Dioxide specifies the pre-condition *Available Material* and the post-condition *Calculated Eco Value*. The *Available Material* pre-condition refers to the resource *Material* and specifies the state *Available*. The *Calculated Eco Value* refers to the resource *Individual Eco Value* and specifies the state *Calculated*. The capability *Issue Carbon Dioxide Certificate* on the other hand, defines the pre-condition *Calculated Eco Value* and the post-condition *Issued Certificate*. *Issued Certificate* relates to the resource *Certificate* and defines its state as *Issued* (cf. figure 5.11).

5.2.5. Place and Time

Figure 5.6 depicts that the fifth CSMM category embodies service entities in terms of place and time. The *Channel* property describes how services are delivered to service consumers electronically or physically. Additionally, this category includes also three enumerations which are utilized by other service properties.

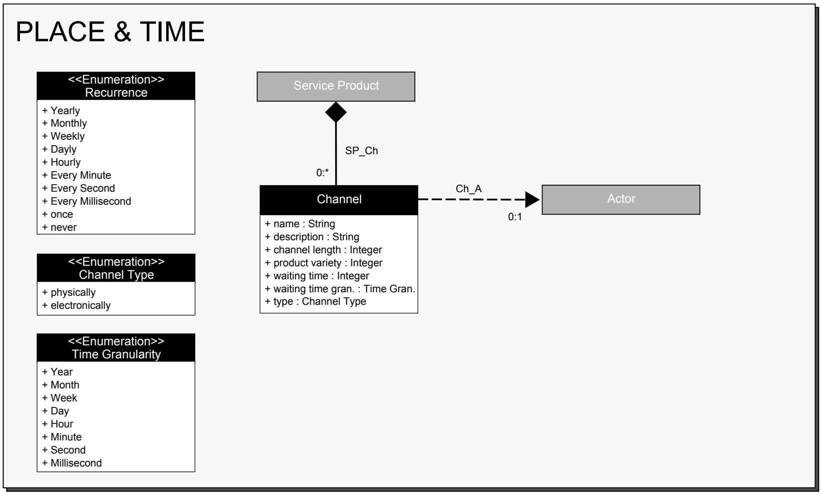


Figure 5.6.: Conceptual Service Meta Model – Place and Time

Channel A *channel* describes any means to deliver services from providers toward consumers. Dumas et al. [35] for example, differentiate between request channels and delivery channels and whether it is a physical or an electronic channel. Lovelock and Wright [71] on the other hand, understand channels in terms of defining *where, when, and how* in terms of delivery. Whereas Mörschel and Höck [76] only particularize a delivery location, Oaks et al. [79] define next to a delivery location a source location. Kotler and Keller [66], finally, define channels in detail including information about channel levels, lot sizes, and waiting time. With CSMM, a channel is `named` and `described` with text. `Channel length` specifies how many middlemen, such as brokers and mediators [18], are between the provider and targeted consumers. `Product variety` tells service consumers how many other services (complementary or unrelated) can be purchased over this channel. `Waiting time (gran.)` specifies how much time passes between a service order and service delivery. `Type` tells whether a channel is electronic or physical. Service products specify none or more channels with the `SP_Ch` relationship. Additionally, channels are either provided by service providers themselves or by another actor. Channel providers are linked to the channel using the `CH_A` relationship.

In the EVC example service, the service is delivered by means of the *TEXO Service Marketplace*. Hence, the channel length is set to *1* as the marketplace is the only middleman between CDE GmbH and its customers. Consequently, the provider link is set to the partner *TEXO Service Marketplace*. The TEXO Service Marketplace as a channel provider offers a wide range of products. Thus, the product variety outlines *150* other service products. The waiting time is set to *160 seconds* and the type to *electronically*.

Time Granularity In order to specify *time*, CSMM uses a elementary mechanism that uses a pair of `Value` and `Time Granularity`. The latter refers to a time metric that specifies *year, month, week, day, hour, minute, second, and millisecond.* In the EVC example service, the channel property defines a waiting time with the value *160* and a granularity of *seconds.*

Recurrence Similar to the time granularity, *recurrence* specifies repeating points in time. Mörschel and Höck [76], for example, define a recurring cycle with the elements *once, per week, and per month.* CSMM goes a step further and defines recurrence with the elements *never, once, every millisecond, every second, every minute, hourly, daily, weekly, monthly, and yearly.* The payment attribute, which is explained in the next CSMM category, utilizes recurrence in order to specify the times when payment is due.

5.2.6. Price

Price and payment are important properties of services marketing and are found prominent in business science and information systems. Figure 5.7 depicts the meta model section with the price and payment properties along with enumerations about currencies, price modifier, and payment instruments. Pricing is an important challenge in service marketing for the following reasons [71]:

- *No ownership of services.*
- *Higher ratio of fixed cost to variable cost.*
- *Many services are hard to evaluate.*
- *Importance of the time factor.*
- *Availability of both, electronic and physical distribution channels.*

Price While Dumas et al. [35], Mörschel and Höck [76], Lovelock and Wright [71], Baida et al. [14], and Grönroos [45] merely acknowledge a price entity, others provide more information. Kotler and Keller [66], for

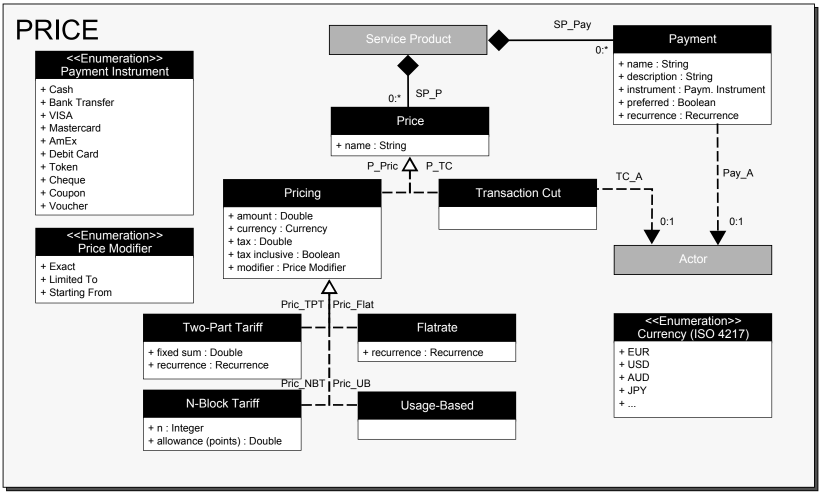


Figure 5.7.: Conceptual Service Meta Model – Price

example, ascribe prices a price objective and a price method. O'Sullivan [100] differentiates between an absolute price, a ranged price, a proportional price, and a dynamic price as well as considers taxes and price modifiers. de Miranda et al. [30] finally, distinguish between four pricing methods. CSMM offers the *price* property as an abstract service property that can be `named`. It is possible to define none or more prices for each service product with the `SP_P` relationship. This property is further refined into the *Transaction Cut* property and the *Pricing* property. It is important to note that capabilities refer to price properties. This makes it possible to specify more than one price for capabilities as well as to link more than one capability to one price (cf. capability property in section 5.2.2).

Pricing The *pricing* property generalizes four different pricing methods [30]: (1) flatrate, (2) usage-based, (3) Two-part tariff, and (4) n-block tariff. The pricing property itself features the following attributes: The `amount` attribute allows specifying a concrete price value. The `currency` attribute, on the other hand, defines the corresponding currency with the ISO 4217 specification that defines an official terminology for currencies. Whereas the `tax` attribute settles the tax percentage, the `tax inclusive` attribute tells whether the specified amount is inclusive or exclusive tax. The `modifier` attribute finally, states whether this price is an exact price, a minimum price, or a maximum price.

A **usage-based pricing** tells potential consumers that this price is due, every time a capability is performed. The EVC service's *Issue Carbon Dioxide Certificate* capability for example features a usage-based *exact* price with an amount of *15*, a *EUR* currency, with *0.19* taxes that are not *included*. **Flatrates**, on the other hand, prescribes that the price is due independent of capability usage. The `recurrence` attribute (cf. section 5.2.5) specifies periods in that the price is due. The EVC service's *Calculate Carbon Dioxide* capability determines a flatrate with an *exact* price and an amount of *300 EUR*, *0.19* taxes that are not *included* and is due *monthly*. A **two-part tariff** combines usages-based and flatrate pricing. The price is due for every capability usage. Additionally, a `fixed sum` is due according to the set `recurrence`. For example, the EVC service's *Issue Carbon Dioxide Certificate* capability specifies a two-part tariff as a second pricing method. In this case the pricing amount is set to *EUR 14*. Additionally, the fixed sum is set to *EUR 100* and the recurrence to *monthly*. An **n-block tariff** finally, is a usage-based pricing method. The difference lies in that after the usage of a certain number `n` the set amount is discounted according to the specified `allowance points`. For example, the EVC service's *Issue Carbon Dioxide Certificate* capability has an n-block tariff pricing method that specifies an amount of *EUR 16* per usage, but after *n=500* invocations an allowance of *0.10* (10 percent) per usage is granted. Figure 5.14 shows these pricings.

Transaction Cut Some business models work without collecting money from service consumers directly. In the insurance industry, for example, insurance brokers find and select appropriate insurance policies for their clients. Brokers are not paid by their clients, however. They rather receive a *courtage* (monetary compensation) from insurance providers. Another example is Internet search engine providers who provide sponsored links according to search terms that search engine consumers may find interesting. Also in this case, the search engine provider is not paid by consumers, but rather from the web link sponsors. The *transaction cost* property may point to another actor of the service description that indicates who pays providers for their services using the `TC_A` relationship.

Payment Lovelock and Wright [71], O'Sullivan [99], Dumas et al. [35], Baida et al. [14], and Momotko et al. [75] discuss payment separately from the price property. It specifies conditions for service consumers *how* and *when* to pay. CSMM defines the payment property with the following attributes. It is possible to `name` and `describe` payments with text. The `instrument` attribute defines *how* to pay. The `Payment Instrument` specifies the following literals: *Cash, Bank Transfer, Mastercard, AmEx, Debit Card, Token, Cheque, Coupon, and Voucher.* In case service providers offer more than one way of payment the `preferred` attribute determines the preferred one. The `recurrence` attribute (cf. section 5.2.5) schedules payments. It is possible to define none or more payment properties for a service product with the `SP_Pay` relationship. In some cases, service providers do not collect payments themselves, but specialized payment collectors, such as Google Checkout and PayPal. In order to specify such a payment collector, service providers reference their payments with another actor using the `Pay_A` relationship. CDE GmbH specifies one *preferred* payment that is due *monthly* via *bank transfer.*

5.2.7. Promotion

CSMM's promotion category targets at building trust between a service provider and potential service consumers, and at attracting additional consumers. The literature refers to certificates, ratings, and test reports in order to establish a trust relationship. Furthermore, price discount systems lower entry boundaries for new consumers. Figure 5.8 outlines CSMM's promotion properties that the following paragraphs detail.

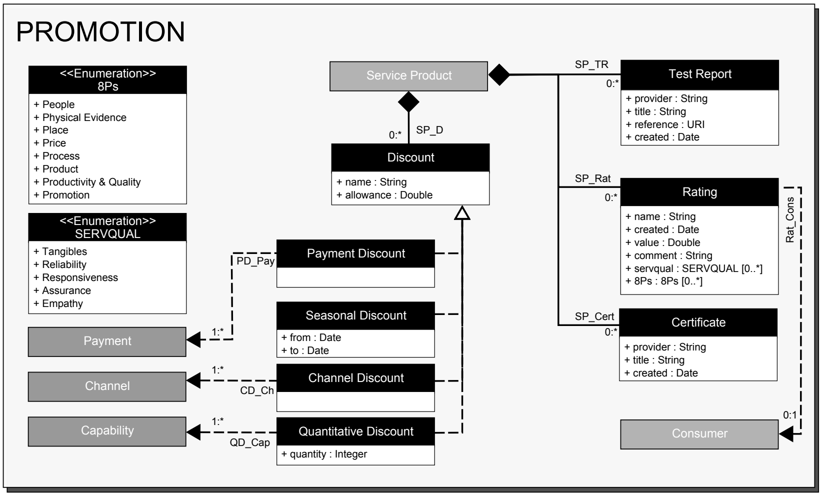


Figure 5.8.: Conceptual Service Meta Model – Promotion

Discount The *discount* property is a marketing instrument that impacts prices in order to attract consumers. Table 5.7 depicts a discount comparison in the literature that differentiates between four aspects. The time aspect refers to the time of payment. This includes early payment discounts as well as seasonal discounts. The way of payment aspect refers to how payment is conducted in terms of payment instruments including cash, credit cards, and coupons. The quantity aspect regards discounts that apply after a certain number of service usages. The location aspect finally, describes discounts that hold for specific locations of service usage.

CSMM provides a generic property and four specific properties for discounts. The discount property can be `named`. The `allowance` attribute

Table 5.7.: Discount comparison

	Time	Way of Payment	Quantity	Location
Kotler & Keller 2007	x	x	x	
Lovelock & Wright 2002				
Dumas et al. 2001	x	x		
O'Sullivan et al. 2005	x	x		x

specifies the percentage that prices will be discounted. This discount is available for all specified prices and circumstances. The `SP_D` relationship ascribes that service products may define none or more discounts. The **payment discount**, however, applies only for selected payment properties with the `PD_Pay` relationship. This is an incentive for consumers to use providers' favorite payment option.

The **seasonal discount** on the other hand, holds only for specific time segments that the date attributes `from` and `to` specify. The **channel discount** refers to the aforementioned location aspect. It specifies discounts in terms of which channels consumers use using the `CD_Ch` relationship. The **quantitative discount** finally, grants allowance depending on how many times consumers *use* services' capabilities. This is specified with the `quantity` attribute and the `CD_Cap` relationship.

The EVC service features a quantitative discount that refers to the *Calculate Carbon Dioxide* capability. The allowance is *0.05* (5 percent) after a usage quantity of *1000*.

Rating Lovelock and Wright [71], Grönroos [45], and O'Sullivan [99] hold information about *rating* in the service marketing literature. Lovelock and

Wright provide the SERVQUAL framework in order to rate services performance in terms of their tangibles, reliability, responsiveness, assurance, and empathy. The authors prescribe that for each aspect, a number between 0 and 5 can be applied to represent consumers' service experiences. Grönroos on the other hand, provides seven different rating aspects that include credibility, security, competence, responsiveness, reliability, access, and personal courtesy. O'Sullivan, finally, provides a more generic yet more formal approach. Rather then defining *what* to rate, they define a rating value for services' overall performance. CSMM defines the rating property with a `rating value` attribute, a `comment` attribute, and a reference to a consumer who owns the rating. Service products can feature unlimited ratings using the `SP_Rat` relationship. In order to refine the rating on certain service aspects, the rating property allows specifying `servqual` literals. Similar to this, it is possible to refine a rating by defining one or more service mix categories (`8Ps`). Additionally, it is possible to link ratings to specific consumers with the `Rat_Cons` relationship.

Certificate *Certifications* are mentioned by Dumas et al. [35] and refer to any document that is provided by a third party which manifests a conformance to a standard. This includes, for example any references to ISO standards. CSMM's certificate property can be `titled` and be provided with certificate `provider` and `creation` information. The `SP_Cert` relationship ascribes none to unlimited certificates for service products.

Test Report This service attribute specifies references to available *test reports*. Test reports might be generated by independent market institutes or test studies in articles, and service products may feature none or more test reports. Similar to the certificate property, test reports can be `titled` and specified with `provider` and `date of creation` details. Furthermore, the `reference` attribute refers to test report original location.

5.2.8. Productivity and Quality

CSMM's Productivity and Quality category embodies eight service properties in order to formalize services' overall productivity, which figure 5.9 shows. These include quality, security, data integrity, confidentially, performance, latency, throughput and dependability. Some evidence for this category is found in business science as well as in information systems. Nevertheless, the majority of indications come from computer science.

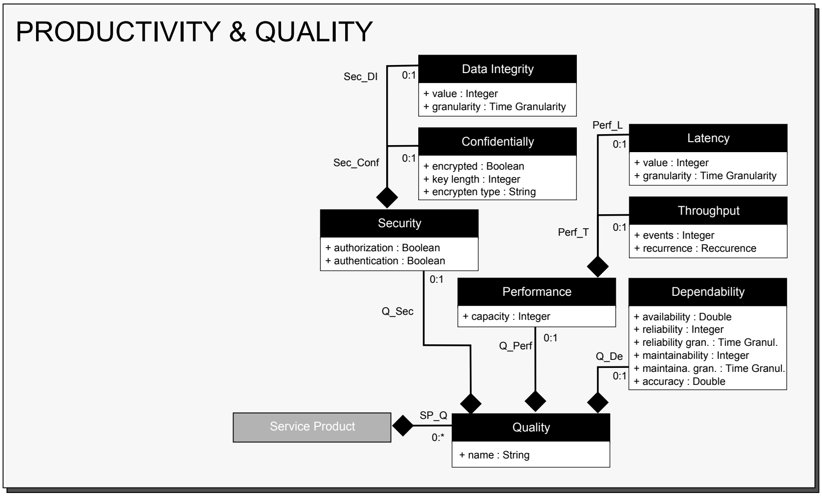


Figure 5.9.: Conceptual Service Meta Model – Productivity and Quality

Quality Formalizing service quality is a complex matter that begins with how to define *service quality*. Computer science offers mainly three directions: (1) service quality in terms of how fast services return results, (2) service quality in terms of security, and (3) service quality in terms of service correctness. Consequently, CSMM summarizes service quality with respect to service performance (`Q_Perf` relationship), security (`Q_Sec` relationship), and dependability (`Q_De` relationship). The quality property can be `named` and capabilities refer to service quality since different services' capabilities possess different qualities. Also, service products may feature none to unlimited quality properties using the `SP_Q` relationship. The EVC service's capability *Calculate Carbon Dioxide* defines the quality *Calculation Quality*.

Dependability The dependability property states services' overall correctness with four aspects: (1) availability, (2) reliability, (3) maintainability, and (4) accuracy.

Dumas et al. [35] as well as O'Sullivan [99] define *availability* as a temporal and a spatial entity. Barbarcci et al. [17], Lee et al. [70], and Avizienis et al. [13] define it as services' readiness for usage. On the other hand, Zeng et al. [147], Brien et al. [95], as well as Giallonardo and Zimeo [41] define availability as the likelihood or proportion that a correct service is accessible and operational. The most formal definition delivers the UPMQoS specification [92]. It distinguishes between the two concepts of *Mean Time To Failure (MTTF)* and *Mean Time To Repair*. The specification then defines services' availability as $\frac{MTTF}{(MTTF+MTTR)} : REAL$. The `availability` attribute follows the UPMQoS definition. The *Calculate Carbon Dioxide* capability has a MTTF of 65 hours and a MTTR of one hour, that results in an availability of *0,985 (98,5%)*.

The definition of *reliability* is divided into two different groups. The first group defines reliability as services' successful execution rate (cf. [7, 75, 147]). The other group defines reliability as the numbers of expected failures for a given time frame (cf. [13, 17, 41, 70, 92, 104]). The `reliability` attribute follows Barbarcci et al.'s [17] definition who define reliability as the MTTF (Mean Time To Failure). As aforementioned, the *Calculate Carbon Dioxide* capability features an MTTF of *65 hours*. Hence the `reliability` attribute is set to *65* and the `reliability granularity` is set to *hour*.

Avizienis et al. [13] ascribe *maintainability* as the *ability to undergo mod-*

ifications and repairs. Barbacci et al. [17] on the other hand, defines maintainability as MTTR (Mean Time To Repair). MTTR defines an average time span for a service to be repaired. The dependability attributes `maintainability` and `maintainability granularity` follows the MTTR definition since it is the most formal one. As aforementioned, the *Calculate Carbon Dioxide* capability features an MTTR of *one hour*. Hence the `maintainability` attribute is set to *1* and the `maintainability granularity` is set to *hour*.

Accuracy in general refers to the correctness of a service. Lee et al. [70] specifies accuracy as the number of errors a service produces during service execution in a given time frame. According to them, this allows consumers to infer the overall service correctness. Thus, the `accuracy` attribute defines the mean success rate for capabilities:

$\frac{SuccessfulExecution}{(SuccessfulExecution+NotSuccessfulExecution)} : REAL.$

For example, past experiences allow calculating the *Calculate Carbon Dioxide* capability an accuracy of *0.995 (99.5%)*.

Performance Computer science agrees on the *performance* definition. UP-MQoS [92], Papaioannou et al. [104], Barbacci et al. [17], Lee et al. [70], as well as Giallonardo and Zimeo [41] define service performance as a combination of latency and throughput which will be discussed below. The performance property optionally defines a throughput property using the `Perf_T` relationship as well as a latency property with the `Perf_L` relationship. Additionally, Barbacci et al. [17], Lee et al. [70], and Giallonardo and Zimeo [41] ascribe performance with a `capacity` attribute that defines the maximum number of parallel service requests.

The EVC service *Calculate Carbon Dioxide* capability outlines a maximum capacity of *20* parallel requests.

Throughput The *throughput* property is defined as a part of capabilities' performance. Whereas Lee et al. [70] and Brien et al. [95] define throughput as the ratio of requests per second, Papaioannou et al. [104] specify throughput in terms of kilobytes per second. Barbacci et al. [17] finally, delineate throughput as the maximum number of events a service can handle in a given time frame. CSMM utilizes this definition for it abstracts from concrete metrics such as kilobytes. The throughput's `event` attribute represent the maximum number of events, and the `recurrence` attribute refers to the given time frame.

The EVC service *Calculate Carbon Dioxide* capability features a throughput of *600* events `every second`, where each request may lead to 30 events.

Latency Amongst others, Barbacci et al. [17] delineate *latency* as services' response window and in general latency refers to the time between a service is triggered and its reaction toward this trigger. CSMM defines the latency property with the `value` attribute and the `granularity` attribute in order to show the maximum time frame that a capability needs to react to a service invocation.

The EVC service *Calculate Carbon Dioxide* capability offers a maximum latency of *5 seconds*.

Security Computer science defines *security* along the notion of confidentially, data integrity as well as access integrity. In CSMM, the security property is further refined by the data integrity and the confidentially property by using the relationships `Sec_DI` and `Sec_Conf`, respectively. The security property itself specifies whether services offer `authorization` and `authentication`.

The EVC service offers *authentication* but no *authorization*.

Data Integrity UPMQoS 2008 [92] defines *data integrity* as the amount of transferred data in a time span without an error. On the other hand, Barbacci et al. [17] along with Lee et al. [70], Avizienis et al. [13], and Giallonardo & Zimeo 2007 [41] see data integrity as the impossibility of unauthorized data modifications. A possible measure for the latter definition offers Barbacci et al. [17]. They measure data integrity in time and resources that are necessary to alter data unauthorized. CSMM formalizes the data integrity property with the two following attributes. `Value` is the time measure that is necessary in order to alter data without proper authorization. The `granularity` attribute specifies the corresponding time granularity.

The EVC service *Calculate Carbon Dioxide* capability with its resources outlines a data integrity of *600 years*.

Confidentially Whereas Barbacci et al. [17] specify data integrity as data resistance toward unauthorized changes, they define *confidentially* as data inaccessibility for unauthorized users. CSMM confidentially property features information about data encryption. The `encrypted` attribute tells whether data is encrypted in the first place. If so, the `encryption type` and its `key length` hold information about the encryption method and its safety level. For example, encryption types include the RSA algorithm or the Advanced Encryption Standards (AES).

Alas the EVC service offers no data encryption.

5.3. Conceptual Service Modeling Notation

Analog to the development of BSMN in section 4.3, this section elaborates on a corresponding notation. The Conceptual Service Modeling Notation (CSMN) intends to support marketing experts and modeling experts while *documenting* and *discussing* conceptual service descriptions, and hence to apply the Conceptual Service Meta Model. Subsection 5.3.4 elaborates on the *CSM result document* for the Eco Calculator example. This section goes through the three step approach, which was introduced in section 2.1.2, in order to develop the notation for CSM.

Table 5.8.: CSMM to UML Mapping and Diff. for CSMM: Product.

#	CSMM	MAPPING	DIFFERENCES
1	**Service Product**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	key	Primitive Datatype	New Datatype
4	description	Primitive Datatype	New Datatype
5	documentation	Primitive Datatype	New Datatype
6	spoken language	Enumeration	New Enumeration
7	written language	Enumeration	New Enumeration
8	version	Primitive Datatype	New Datatype
9	created	Primitive Datatype	New Datatype
10	updated	Primitive Datatype	New Datatype
11	next update	Primitive Datatype	New Datatype
12	type	Enumeration	New Enumeration
13	automation	Enumeration	New Enumeration
14	composition	Enumeration	New Enumeration
15	customizable	Enumeration	New Datatype
16	**Benefit**	Class	New UML Class
17	name	Primitive Datatype	N/A
18	description	Primitive Datatype	New Datatype
19	**Terms of Use**	Class	New UML Class
20	payment condition	Primitive Datatype	New Datatype
21	delivery condition	Primitive Datatype	New Datatype
22	**Classification**	Class	New UML Class
23	value	Primitive Datatype	New Datatype
24	system	Primitive Datatype	New Enumeration
25	**Lang ISO (639-2)**	Enumeration	New Enumeration
26	eng	Enumeration Literal	New Enumeration Literal
27	fra	Enumeration Literal	New Enumeration Literal
28	ger	Enumeration Literal	New Enumeration Literal
29	**Service Type**	Enumeration	New Enumeration
30	Core Service	Enumeration Literal	New Enumeration Literal
31	Supporting Service	Enumeration Literal	New Enumeration Literal
32	Enhancing Service	Enumeration Literal	New Enumeration Literal
33	**Automation Level**	Enumeration	New Enumeration
34	Fully Automated	Enumeration Literal	New Enumeration Literal
35	Partially Automated	Enumeration Literal	New Enumeration Literal
36	Manual	Enumeration Literal	New Enumeration Literal
37	**Classification System**	Enumeration	New Enumeration
38	NAICS	Enumeration Literal	New Enumeration Literal
39	Ecl@ss	Enumeration Literal	New Enumeration Literal
40	UNSPSC	Enumeration Literal	New Enumeration Literal
41	Nice Classification	Enumeration Literal	New Enumeration Literal

#	CSMM	MAPPING	DIFFERENCES
42	**Composition Level**	Enumeration	New Enumeration
43	intermediary Service	Enumeration Literal	New Enumeration Literal
44	Final Service	Enumeration Literal	New Enumeration Literal
45	**SP_SP**	Association	New Property
46			memberEnd.type = Service Product
47	**SP_B**	Association (Composite Agg.)	New Property
48			memberEnd.type = Benefit
49	**SP_ToU**	Association (Composite Agg.)	New Property
50			memberEnd.type = Terms of Use
51			memberEnd.upper = 1
52	**SP_C**	Association (Composite Agg.)	New Property
53			memberEnd.type = Classification

5.3.1. Step 1: Definition of Integration Meta Model

The first step is to establish a mapping between CSMM elements (cf. tables 5.8—5.15) and UML meta model elements. This step clarifies how to represent domain-specific elements with UML elements. Four main areas for mapping exist: (1) Classes & Properties, (2) Enumerations & Literals, (3) Associations & Composite Aggregations, and (4) Generalizations.

For example table 5.8 in line #1 shows that the element `Service Product` maps to `UML Class`. Its attributes *name, key, description, documentation, version, created, updated, and next update* map to the UML meta model element `Primitive Datatype`. This holds also true for all service properties in the tables 5.8–5.15.

Similar to classes and properties, the next step tells to map enumerations and their literals to UML meta model elements. The element `Service Type` in table 5.8 in line #29, for example, corresponds to the UML meta model element `Enumeration`. The literals *Core Service, Supporting Service, and Enhancing Service* map to UML's `Enumeration Literals`. Tables 5.8–5.15 show the mapping of other enumeration elements.

Relationships between CSMM elements are also part of this mapping step. It is important to differentiate between three types of relationships: (1) association, (2) composite aggregation, and (3) generalization. An *association* ascribes a semantic relationship between two or more elements [87]. While the *composite aggregation* is a specialized aggregation, it defines a relationship between two elements. It tells that a connected element is not independent of the first element and may only exists in coexistence [87]. *Generalizations*, on the other hand, define a relationship between two elements, where one element is the generic and the other specific. The specific element inherits the generic's properties [87].

Table 5.10 shows the mapping for CSMM's people category (cf. subsection 5.2.3). Line #35 shows the mapping between the `A_Rig` relationship towards a UML `Association`. It acknowledges the fact that actors may hold copyrights and property rights. However, right elements do no depend on the actor element. Rather, the `Right` element *depends* on the `Service Product` element. Line #26 discloses the `SP_Rig` relationship as a *Composite Aggregation* with the meaning that rights belong to a service product. The lines #40–42 finally show three relationship mappings to the UML *Generalization* element. These relationships tell that CSMM's `Actor` element is the *generic* element and the elements `Provider, Consumer`, and `Partner` are specific elements. This implies that the latter elements inherit all features of the `Actor` element [87].

Whereas the previous paragraphs elaborated only exemplarily on the mapping, tables 5.8–5.15 provide a detailed mapping between elements of CSMM and the UML meta model. This mapping serves as input for step 2.

5.3.2. Step 2: Meta Model Comparison

With the availability of the mapping between CSMM and the UML meta model from step 1, this step outlines differences between the two meta models. Each discovered discrepancy needs to be considered for the UML Profile generation in order to constrain or to amplify UML's meta model. The tables 5.8–5.15 show deviances for all mappings that the following paragraphs explain exemplarily for classes and properties, enumerations and literals, and associations, composite aggregations and generalizations.

Classes & Properties Classes and their properties that were identified in step 1 were mapped to the UML meta model. For example in table 5.11, the `Resource` element in line #1 is mapped to UML Class. Since the UML meta model embodies no such element, the Resource is marked as *New UML Class*. In contrary, the `Resource`'s name attribute (line #2) was mapped to the existing UML Class property *name*, and hence, equals the UML meta model. However, the `Resource`'s property *description* (line #3) may not be directly mapped and is marked with *New Datatype*. The tables 5.8–5.15 show that the identification of differences between classes and properties is done in the same manner for all CSMM elements.

Enumerations & Literals Likewise, enumerations and their literals were mapped to the UML meta model. E.g., table 5.12 in line #29 shows the mapping between the CSMM's `Channel Type` and the `UML Enumeration` that is marked as a *New Enumeration* for it is nonexistent in the UML meta model. Furthermore, the mappings in lines #30–31 for the literals, *Physically* and *Electronically* do not exist in the UML meta model, and in consequence, are marked as *New Enumeration Literals*. This mapping is similar to the other enumerations, such as `Recurrence` and `Time Granularity`.

Associations, Composite Aggregations, and Generalizations Lastly, CSMM's relationships need to be mapped to the UML meta model. CSMM outlines 46 relationships for interconnecting service properties that are distinguished between associations, aggregations, and generalizations.

For example, the relationship `TC_A` that is mapped to an UML `Associat-`

ion tells that a `Transaction Cut` element links to none or to exact one `Actor` (cf. table 5.13 in line #53–55). However, the difference between CSMM and UML is that in case of the CSMM, the `Transaction Cut` may be only associated with an `Actor`, whereas the UML `Association` defines its `memberEnd.type` with any related element, and thus, the `member-End.type` is marked with *memberEnd.type = Actor*. Another difference between a UML `Association` and `TC_A` is that the association defines its lower limit with *0* and its upper limit with *unlimited* [87]. In case of the `TC_A` relationship, the upper limit is *1* (cf. figure 5.7). Hence, line #55 acknowledges this differences in that it sets the `memberEnd.upper = 1`.

This is analog for all relationships that map to UML `Composite Aggre-gations`, such as the `SP_P` relationship in line #48 that ascribes that `Service Product` elements define none or more `Price` elements.

Lines #56–61 mark all relationships that map to UML `Generalization` elements with *New Generalization*. The tables 5.8–5.15 show that the identification of differences between associations, aggregations, and generalizations is done in the same manner for all CSMM relationships.

5.3.3. Step 3: Integration Meta Model Transformation

The last step takes into account the discovered differences in step 2 for the UML Profile with *transformation rules*. Eleven rules (cf. [40]) are the basis. This subsection goes through the rules 1–4, 6, and 8 one by one for classes, properties, associations, enumerations, and generalizations. Other rules are skipped for they are not necessary for the CSMN.

Rule 1: *(one Stereotype for each class)* Rule 1 aims at new classes. The `Channel` element in table 5.12's line #1 for example is marked as a *New UML Class*. Rule 1 ascribes that each new class needs to be represented with a new `Stereotype` in the UML Profile definition which in turn is mapped to the metaclass `UML Class`. This is repeated for all elements that are marked with *New UML Class* (cf. tables 5.8–5.15).

Rule 2: *(one Tagged Value for each new property)* Properties comprise *attributes* as well as *associations* and *composite aggregations*. Tagged values consist of a `name` and a `type`. In step 2 discovered new properties will be represented with a tagged value and a corresponding OCL expression.

For example, the composite aggregation relationship `SP_Ch` in line #32 in table 5.12 is in fact a property of the `Service Product` element. This property is represented with the following *tagged value* `SP_Ch:Channel`. Furthermore, next to the tagged value, an OCL expression is added that enforces the different memberEnd.type:
`self.SP_Ch -> isStereotyped(Channel)`.

Rule 3: *(one OCL constraint if elements' lower bound are higher than the lower bound of the referenced UML property)* As mentioned in step 2, UML relationships define their lower limits with *0* and their upper limits with *unlimited*. This is different for some relationships in CSMM. The relationship `Con_R` specifies that that each `Condition` element links exactly *one* `Resource` element (cf. table 5.11, line #24). Figure 5.5 shows its multiplicity as *1:1*. Hence, the lower bound of this relationship must be raised to 1. The OCL expression `self.Con_R.size >= 1` accomplish this. In consequence, an instance of the UML Profile validates only if each *Condition* element links to at least one `Resource` element.

Rule 4: *(one OCL constraint if elements' higher bound are lower than the higher bound of the referenced UML property)* Complementary to rule 3, rule 4 addresses the higher bounds of properties. Sticking to the aforementioned example of the `Con_R` relationship, the higher bound needs to be set from *unlimited* to *1*. The second OCL expression `self.Con_R.size <= 1` accomplish this. In consequence, an instance of the UML Profile validates only if each *Condition* element links to at the most to one `Resource` element.

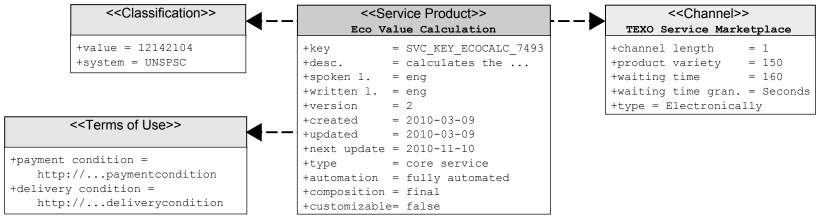


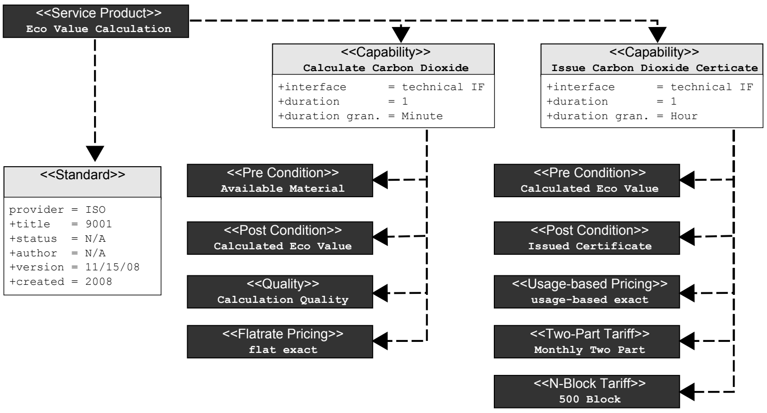
Figure 5.10.: Conceptual Service Diagram for *Eco Value Calculation* Service: Product and Place & Time

Rule 6: *(one Enumeration for each new enumeration with new literals)* Each of the CSMM's enumerations with their literals are acknowledged with a UML Enumeration. The `Channel Type` element with its literals in lines #29–31 in table 5.12 for example are represented with a UML `enumeration` and two UML `enumeration literals` in the UML Profile definition.

Rule 8: *(duplicate properties of each general class for each of their specialized classes)* In order to represent the UML `Generalization` relationship, all properties of generic classes must be duplicated for their specialized classes. This applies for example to the `Pric_Flat` generalization in line #59 of table 5.13. The `Pricing` element specifies the following attributes: *amount, currency, tax, tax inclusive,* and *modifier*. The `Flatrate` element specifies just one attribute (cf. line # 17–18). For the UML Profile definition, however, all `Pricing` attributes are duplicated and also specified for the `Flatrate` element.

5.3.4. CSM Result Diagram

It is possible to use the generated UML Profile CSMN with every UML-compliant UML Tool, including the Eclipse UML2 Toolset [1]. Chapter 7 introduces the Conceptual Service Modeling Tool that implements this modeling notation.



<<Service Product>> Eco Value Calculation
<<Capability>> Calculate Carbon Dioxide
+interface = technical IF
+duration = 1
+duration gran. = Minute
<<Capability>> Issue Carbon Dioxide Certicate
+interface = technical IF
+duration = 1
+duration gran. = Hour
<<Standard>>
provider = ISO
+title = 9001
+status = N/A
+author = N/A
+version = 11/15/08
+created = 2008
<<Pre Condition>> Available Material
<<Post Condition>> Calculated Eco Value
<<Quality>> Calculation Quality
<<Flatrate Pricing>> flat exact
<<Pre Condition>> Calculated Eco Value
<<Post Condition>> Issued Certificate
<<Usage-based Pricing>> usage-based exact
<<Two-Part Tariff>> Monthly Two Part
<<N-Block Tariff>> 500 Block

Figure 5.11.: Conceptual Service Diagram for *Eco Value Calculation* Service: Process

Figures 5.10–5.15 show the conceptual service diagram for the *Eco Value Calculation* service as the result documents for CSM, and hence, they show the application of CSMN. The whole diagram is divided into these six figures for practical reasons. Listing A.9 shows the complete corresponding XML code for this diagram.

Figure 5.10 depicts the product as well as the place and time category. The root of the conceptual service description is the **service product** property *Eco Value Calculation*. For identification, the key *SVC_KEY_ECOCALC_7493* is provided and further described as *calculates the carbon dioxide footprint for any given material*. The service is only offered in one language: *eng*. The EVC service version is *2*, was created on *2010-03-09*, last updated on *2010-03-09*, and will receive the next update at *2010-11-01*. From the perspective of the CDE GmbH, the EVC service is rather a *core service* as well as a *final service*. Customers have no influence on the service's customization.

The service features one **classification** using the *UNSPSC* system. Its value *12142104* states that the service belongs to the class *Carbon dioxide* *gas CO2*. The **terms of use** for the EVC service points to *http://62.52.175.245:8080/texo/ services/ecocalc/paymentcondition* and *http ://62.52.175.245:8080/texo/services/ecocalc/deliverycondition*, respectively.

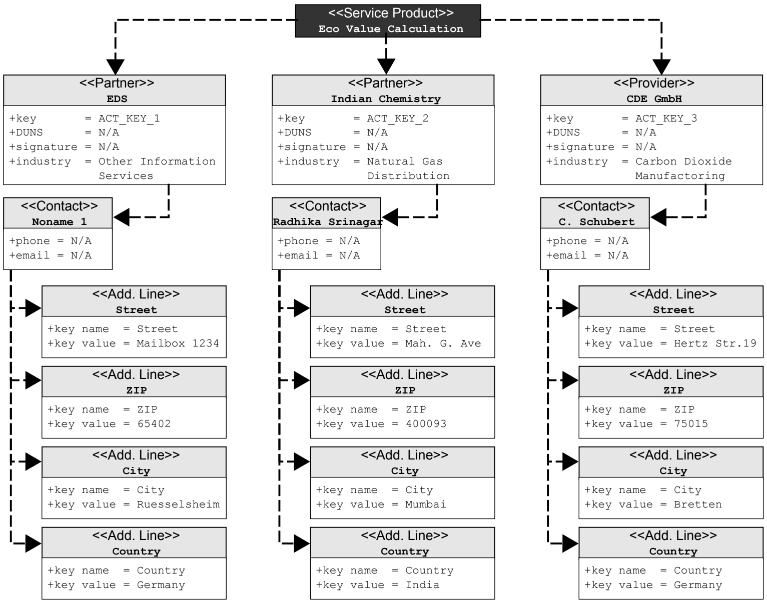


Figure 5.12.: Conceptual Service Diagram for *Eco Value Calculation* Service: People

Finally, the conceptual service description defines one **channel** *TEXO Service Marketplace* with a channel length of *1* since only the service market place is between the provider and possible consumers. The *electronically* channel overall features *150* products and has a maximal waiting time of *160 seconds*.

Figure 5.11 shows the part of the CSM diagram for properties in the process category. The left hand side shows that the EVC service brings along the *ISO 9001* **standard**, which indicates a quality management system of CDE GmbH.

Two **capabilities** are defined in the conceptual service description. The *Calculate Carbon Dioxide* capability has a *technical interface* and a duration of *1 Minute*. The capability owns the pre condition *Available Material* and the post condition *Calculated Eco Value* which figure 5.13 shows in detail. Also, the capability links to the quality *Calculation Quality* (cf. figure 5.15) and to the pricing *flat exact* (cf. figure 5.14). The second capability *Issue Carbon Dioxide Certificate* also features a *technical interface* with a duration of *1 Hour*. It owns the pre condition *Calculated Eco Value* and the post condition *Issued Certificate* (cf. figure 5.13). On top of that, the capability links to three available pricings: *usage-based exact, monthly two part, and 500 block*, which figure 5.14 details.

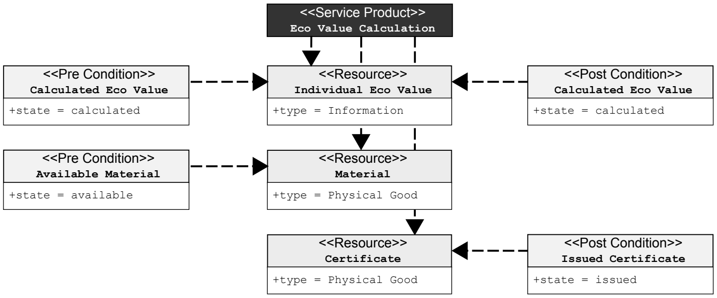


Figure 5.13.: Conceptual Service Diagram for *Eco Value Calculation* Service: Physical Evidence

Figure 5.12 depicts service properties that belong in the people category. It shows that the service description defines three *actors*. The **provider** in the right hand side of the figure is the *CDE GmbH*. The provider owns the unique key *ACT_KEY_3* and is in the *Carbon Dioxide Manufactoring* industry. For the provider is the contact *C. Schubert* defined. Furthermore, the two partners *EDS* and *Indian Chemistry* are also part of the service product.

Service properties that belong to the physical evidence category are shown in figure 5.13. The EVC service defines three **resources**. The *Individual*

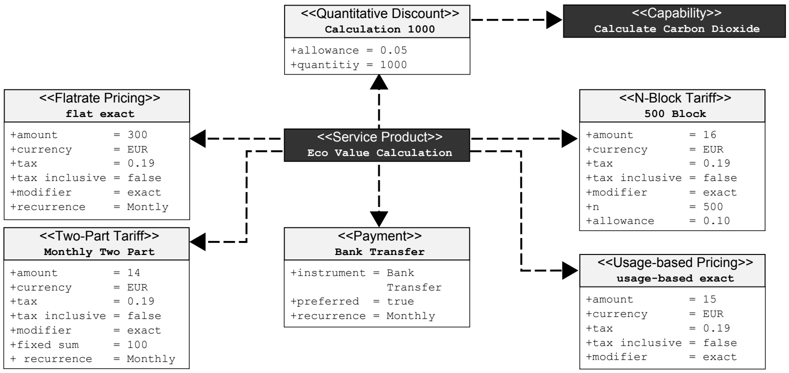


Figure 5.14.: Conceptual Service Diagram for *Eco Value Calculation* Service: Price and Promotion

Eco Value is an *information* type resource. The *Material* and the *Certificate* are both of type *Physical Good*.

Furthermore, four **condition** exists, which were discussed along with the process properties (cf. figure 5.11). The pre condition *Calculated Eco Value* links to the *Individual Eco Value* resources and has the state *calculated*. The *Available Material* pre condition refers to the *Material* resource and depicts a state *available*. The post condition *Calculated Eco Value* links as well to the *Individual Eco Value* resources and outlines its state as *calculated*. The post condition *Issued Certificate* finally, refers to the *Certificate* with the state *issued*.

Figure 5.14 shows price and promotion properties. The **payment** property *Bank Transfer* indicates this is the *preferred* payment and that payments are due *monthly* via *Bank Transfer*.

The service product also defines four different **pricings** that were previously discussed with the capabilities (cf. figure 5.14). The *usage-based exact* is a Usage-based pricing that specifies an *exact* amount of *EUR 15* with an *exclusive* tax of *19%*. The *flat exact*, on the other hand, is a Flatrate Pricing, which indicates an exact amount of *EUR 300* that is due *Monthly*.

The *Monthly Two Part* is a Two-Part Tariff pricing and allows an amount of *EUR 14* for service usage as well as a *monthly* fixed sum of *EUR 100*. The N-Block Tariff pricing *500 block* sets an exact amount of *EUR 16* and grants an allowance of *10%* after the usage of 500 times.

The upper part of figure 5.14 shows the **Quantitative Discount** *Calculation 1000* gives an allowance of *5%* after the usage of *1000* times of the *Calculate Carbon Dioxide* capability.

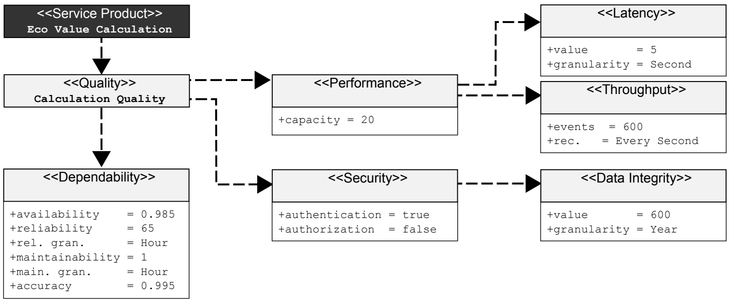


Figure 5.15.: Conceptual Service Diagram for *Eco Value Calculation* Service: Productivity & Quality

Figure 5.15 shows productivity and quality service properties. The EVC service defines the **quality** *Calculation Quality*. The capability *Calculate Carbon Dioxide* refers to this quality property. Hence an assertion can be made about the capabilities quality in terms of its dependability, performance, and security.

The quality owns a **dependability** with an availability of *98.5%*. This is calculated with the proportion between the mean time to failure and mean time to repair (cf. subsection 5.2.8).

Furthermore, the quality property outlines a **performance** with a capacity of *20* parallel requests, a **latency** of *5 seconds* and a **throughput** of *600 events per second*. It also depicts a **security** that provides *authentication* but no *authorization* as well as a **data integrity**, which depicts *600 years* that are necessary in order to alter the data without proper authentication.

Table 5.9.: CSMM to UML Mapping and Diff. for CSMM: Process.

#	CSMM	MAPPING	DIFFERENCES
1	**Capability**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	interface	Enumeration	New Enumeration
5	duration	Primitive Datatype	New Datatype
6	duration granularity	Enumeration	New Enumeration
7	**Standard**	Class	New UML Class
8	provider	Primitive Datatype	New Datatype
9	title	Primitive Datatype	New Datatype
10	status	Primitive Datatype	New Datatype
11	author	Primitive Datatype	New Datatype
12	version	Primitive Datatype	New Datatype
13	created	Primitive Datatype	New Datatype
14	**Interface Type**	Enumeration	New Enumeration
15	Personnel Interface	Enumeration Literal	New Enumeration Literal
16	Web Interface	Enumeration Literal	New Enumeration Literal
17	Technical Interface	Enumeration Literal	New Enumeration Literal
18	**Cap_P**	Association	New Property
19			memberEnd.type = Price
20	**Cap_Q**	Association	New Property
21			memberEnd.type = Price
22			memberEnd.upper = 1
23	**SP_Cap**	Association (Composite Agg.)	New Property
24			memberEnd.type = Capability
25			memberEnd.lower = 1
26	**SP_St**	Association (Composite Agg.)	New Property
27			memberEnd.type = Standard
28	**Cap_Con**	Association (Composite Agg.)	New Property
29			memberEnd.type = Standard

Table 5.10.: CSMM to UML Mapping and Diff. for CSMM: People.

#	CSMM	MAPPING	DIFFERENCES
1	**Right**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	type	Enumeration	New Enumeration
5	**Actor**	Class	New UML Class
6	name	Primitive Datatype	N/A
7	description	Primitive Datatype	New Datatype
8	key	Primitive Datatype	New Datatype
9	DUNS	Primitive Datatype	New Datatype
10	signature	Primitive Datatype	New Datatype
11	industry	Primitive Datatype	New Datatype
12	**Contact**	Class	New UML Class
13	person name	Primitive Datatype	New Datatype
14	description	Primitive Datatype	New Datatype
15	phone	Primitive Datatype	New Datatype
16	email	Primitive Datatype	New Datatype
17	**Address Line**	Class	New UML Class
18	key name	Primitive Datatype	New Datatype
19	key value	Primitive Datatype	New Datatype
20	**Provider**	Class	New UML Class
21	**Consumer**	Class	New UML Class
22	**Partner**	Class	New UML Class
23	**Right Type**	Enumeration	New Enumeration
24	Copyright	Enumeration Literal	New Enumeration Literal
25	Property Right	Enumeration Literal	New Enumeration Literal
26	**SP_Rig**	Association (Composite Agg.)	New Property
27			memberEnd.type = Right
28	**SP_A**	Association (Composite Agg.)	New Property
29			memberEnd.type = Actor
30	**SP_Co**	Association (Composite Agg.)	New Property
31			memberEnd.type = Contact
32	**Co_Add**	Association (Composite Agg.)	New Property
33			memberEnd.type = Address Line
34			memberEnd.lower = 1
35	**A_Rig**	Association	New Property
36			memberEnd.type = Right

#	CSMM	MAPPING	DIFFERENCES
37	**Rig_R**	Association	New Property
38			memberEnd.type = Resource
39			memberEnd.lower = 1
40	**A_Prov**	Generalization	New Generalization
41	**A_Cons**	Generalization	New Generalization
42	**A_Part**	Generalization	New Generalization

Table 5.11.: CSMM to UML Mapping and Diff. for CSMM: Physical Evidence.

#	CSMM	MAPPING	DIFFERENCES
1	**Resource**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	type	Enumeration	New Enumeration
5	**Condition**	Class	New UML Class
6	name	Primitive Datatype	N/A
7	description	Primitive Datatype	New Datatype
8	state	Primitive Datatype	New Datatype
9	**Pre Condition**	Class	New UML Class
10	**Post Condition**	Class	New UML Class
11	**Resource Type**	Enumeration	New Enumeration
12	Physical Good	Enumeration Literal	New Enumeration Literal
13	Service	Enumeration Literal	New Enumeration Literal
14	Information	Enumeration Literal	New Enumeration Literal
15	Media	Enumeration Literal	New Enumeration Literal
16	Personnel	Enumeration Literal	New Enumeration Literal
17	Capability	Enumeration Literal	New Enumeration Literal
18	Experience	Enumeration Literal	New Enumeration Literal
19	Monetary	Enumeration Literal	New Enumeration Literal
20	**SP_R**	Association (Composite Agg.)	New Property
21			memberEnd.type = Resource
22	**Con_R**	Association	New Property
23			memberEnd.type = Resource
24			memberEnd.lower = 1
25			memberEnd.upper = 1
26	**Con_Pre**	Generalization	New Generalization
27	**Con_Post**	Generalization	New Generalization

Table 5.12.: CSMM to UML Mapping and Diff. for CSMM: Place & Time.

#	CSMM	MAPPING	DIFFERENCES
1	**Channel**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	channel length	Primitive Datatype	New Datatype
5	product variety	Primitive Datatype	New Datatype
6	waiting time	Primitive Datatype	New Datatype
7	waiting time gran.	Enumeration	New Enumeration
8	type	Enumeration	New Enumeration
9	**Recurrence**	Enumeration	New Enumeration
10	Yearly	Enumeration Literal	New Enumeration Literal
11	Monthly	Enumeration Literal	New Enumeration Literal
12	Weekly	Enumeration Literal	New Enumeration Literal
13	Daily	Enumeration Literal	New Enumeration Literal
14	Hourly	Enumeration Literal	New Enumeration Literal
15	Every Minute	Enumeration Literal	New Enumeration Literal
16	Every Second	Enumeration Literal	New Enumeration Literal
17	Every Millisecond	Enumeration Literal	New Enumeration Literal
18	once	Enumeration Literal	New Enumeration Literal
19	never	Enumeration Literal	New Enumeration Literal
20	**Time Granularity**	Enumeration	New Enumeration
21	Year	Enumeration Literal	New Enumeration Literal
22	Month	Enumeration Literal	New Enumeration Literal
23	Week	Enumeration Literal	New Enumeration Literal
24	Day	Enumeration Literal	New Enumeration Literal
25	Hour	Enumeration Literal	New Enumeration Literal
26	Minute	Enumeration Literal	New Enumeration Literal
27	Second	Enumeration Literal	New Enumeration Literal
28	Millisecond	Enumeration Literal	New Enumeration Literal
29	**Channel Type**	Enumeration	New Enumeration
30	Physically	Enumeration Literal	New Enumeration Literal
31	Electronically	Enumeration Literal	New Enumeration Literal
32	**SP_Ch**	Association (Composite Agg.)	New Property
33			memberEnd.type = Channel
34	**Ch_A**	Association	New Property
35			memberEnd.type = Actor
36			memberEnd.upper = 1

Table 5.13.: CSMM to UML Mapping and Diff. for CSMM: Price.

#	CSMM	MAPPING	DIFFERENCES
1	**Payment**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	description	Primitive Datatype	New Datatype
4	instrument	Enumeration	New Enumeration
5	preferred	Primitive Datatype	New Datatype
6	recurrence	Enumeration	New Enumeration
7	**Price**	Class	New UML Class
8	name	Primitive Datatype	N/A
9	**Transaction Cut**	Class	New UML Class
10	**Pricing**	Class	New UML Class
11	amount	Primitive Datatype	New Datatype
12	currency	Enumeration	New Enumeration
13	tax	Primitive Datatype	New Datatype
14	tax inclusive	Primitive Datatype	New Datatype
15	modifier	Enumeration	New Enumeration
16	**Usage-Based**	Class	New UML Class
17	**Flatrate**	Class	New UML Class
18	recurrence	Enumeration	New Enumeration
19	**Two-Part Tariff**	Class	New UML Class
20	fixed sum	Primitive Datatype	New Datatype
21	recurrence	Enumeration	New Enumeration
22	**N-Block Tariff**	Class	New UML Class
23	n	Primitive Datatype	New Datatype
24	allowance (points)	Primitive Datatype	New Datatype
25	**Currency (ISO 4217)**	Enumeration	New Enumeration
26	EUR	Enumeration Literal	New Enumeration Literal
27	USD	Enumeration Literal	New Enumeration Literal
28	AUD	Enumeration Literal	New Enumeration Literal
29	JPY	Enumeration Literal	New Enumeration Literal
30	…	Enumeration Literal	New Enumeration Literal
31	**Payment Instrument**	Enumeration	New Enumeration
32	Cash	Enumeration Literal	New Enumeration Literal
33	Bank Transfer	Enumeration Literal	New Enumeration Literal
34	VISA	Enumeration Literal	New Enumeration Literal
35	Mastercard	Enumeration Literal	New Enumeration Literal
36	AmEx	Enumeration Literal	New Enumeration Literal
37	Debit Card	Enumeration Literal	New Enumeration Literal
38	Token	Enumeration Literal	New Enumeration Literal
39	Cheque	Enumeration Literal	New Enumeration Literal
40	Coupon	Enumeration Literal	New Enumeration Literal
41	Voucher	Enumeration Literal	New Enumeration Literal

#	CSMM	MAPPING	DIFFERENCES
42	**Price Modifier**	Enumeration	New Enumeration
43	Exact	Enumeration Literal	New Enumeration Literal
44	Limited to	Enumeration Literal	New Enumeration Literal
45	Starting from	Enumeration Literal	New Enumeration Literal
46	**SP_Pay**	Association (Composite Agg.)	New Property
47			memberEnd.type = Payment
48	**SP_P**	Association (Composite Agg.)	New Property
49			memberEnd.type = Price
50	**Pay_A**	Association	New Property
51			memberEnd.type = Actor
52			memberEnd.upper = 1
53	**TC_A**	Association	New Property
54			memberEnd.type = Actor
55			memberEnd.upper = 1
56	**P_TC**	Generalization	New Generalization
57	**P_Pric**	Generalization	New Generalization
58	**Pric_UB**	Generalization	New Generalization
59	**Pric_Flat**	Generalization	New Generalization
60	**Pric_TPT**	Generalization	New Generalization
61	**Pric_NBT**	Generalization	New Generalization

Table 5.14.: CSMM to UML Mapping and Diff. for CSMM: Promotion.

#	CSMM	MAPPING	DIFFERENCES
1	**Test Report**	Class	New UML Class
2	provider	Primitive Datatype	New Datatype
3	title	Primitive Datatype	New Datatype
4	reference	Primitive Datatype	New Datatype
5	created	Primitive Datatype	New Datatype
6	**Rating**	Class	New UML Class
7	name	Primitive Datatype	N/A
8	created	Primitive Datatype	New Datatype
9	value	Primitive Datatype	New Datatype
10	comment	Primitive Datatype	New Datatype
11	servqual	Enumeration	New Enumeration
12	8Ps	Enumeration	New Enumeration
13	**Rating**	Class	New UML Class
14	provider	Primitive Datatype	New Datatype
15	title	Primitive Datatype	New Datatype
16	created	Primitive Datatype	New Datatype
17	**Discount**	Class	New UML Class
18	name	Primitive Datatype	N/A
19	allowance	Primitive Datatype	New Datatype
20	**Payment Discount**	Class	New UML Class
21	**Seasonal Discount**	Class	New UML Class
22	from	Primitive Datatype	New Datatype
23	to	Primitive Datatype	New Datatype
24	**Channel Discount**	Class	New UML Class
25	**Quantitative Discount**	Class	New UML Class
26	quantity	Primitive Datatype	New Datatype
27	**8Ps**	Enumeration	New Enumeration
28	People	Enumeration Literal	New Enumeration Literal
29	Physical Evidence	Enumeration Literal	New Enumeration Literal
30	Place	Enumeration Literal	New Enumeration Literal
31	Price	Enumeration Literal	New Enumeration Literal
32	Process	Enumeration Literal	New Enumeration Literal
33	Product	Enumeration Literal	New Enumeration Literal
34	Productivity & Quality	Enumeration Literal	New Enumeration Literal
35	Promotion	Enumeration Literal	New Enumeration Literal
36	**SERVQUAL**	Enumeration	New Enumeration
37	Tangibles	Enumeration Literal	New Enumeration Literal
38	Reliability	Enumeration Literal	New Enumeration Literal
39	Responsiveness	Enumeration Literal	New Enumeration Literal
40	Assurance	Enumeration Literal	New Enumeration Literal
41	Empathy	Enumeration Literal	New Enumeration Literal

#	CSMM	MAPPING	DIFFERENCES
42	**SP_TR**	Association (Composite Agg.)	New Property
43			memberEnd.type = Test Report
44	**SP_R**	Association (Composite Agg.)	New Property
45			memberEnd.type = Rating
46	**SP_Cert**	Association (Composite Agg.)	New Property
47			memberEnd.type = Certificate
48	**SP_D**	Association (Composite Agg.)	New Property
49			memberEnd.type = Discount
50	**PD_Pay**	Association	New Property
51			memberEnd.type = Payment
52			memberEnd.lower = 1
53	**CD_Ch**	Association	New Property
54			memberEnd.type = Channel
55			memberEnd.lower = 1
56	**QD_Cap**	Association	New Property
57			memberEnd.type = Capability
58			memberEnd.lower = 1
59	**R_Cons**	Association	New Property
60			memberEnd.type = Consumer
61			memberEnd.upper = 1
62	**D_PD**	Generalization	New Generalization
63	**D_SD**	Generalization	New Generalization
64	**D_CD**	Generalization	New Generalization
65	**D_QD**	Generalization	New Generalization

Table 5.15.: CSMM to UML Mapping and Differences for CSMM: Productivity & Quality.

#	CSMM	MAPPING	DIFFERENCES
1	**Quality**	Class	New UML Class
2	name	Primitive Datatype	N/A
3	**Dependability**	Class	New UML Class
4	availability	Primitive Datatype	New Datatype
5	reliability	Primitive Datatype	New Datatype
6	reliability gran.	Enumeration	New Enumeration
7	maintainability	Primitive Datatype	New Datatype
8	maintainability gran.	Enumeration	New Enumeration
9	accuracy	Primitive Datatype	New Datatype
10	**Performance**	Class	New UML Class
11	capacity	Primitive Datatype	New Datatype
12	**Latency**	Class	New UML Class
13	value	Primitive Datatype	New Datatype
14	granularity	Enumeration	New Enumeration
15	**Throughput**	Class	New UML Class
16	events	Primitive Datatype	New Datatype
17	recurrence	Enumeration	New Enumeration
18	**Security**	Class	New UML Class
19	authorization	Primitive Datatype	New Datatype
20	authentication	Primitive Datatype	New Datatype
21	**Data Integrity**	Class	New UML Class
22	value	Primitive Datatype	New Datatype
23	granularity	Enumeration	New Enumeration
24	**Confidentially**	Class	New UML Class
25	encrypted	Primitive Datatype	New Datatype
26	key length	Primitive Datatype	New Datatype
27	encryption type	Primitive Datatype	New Datatype
28	**SP_Q**	Association (Composite Agg.)	New Property
29			memberEnd.type = Quality
30	**Q_De**	Association (Composite Agg.)	New Property
31			memberEnd.type = Dependability
32			memberEnd.upper = 1
33	**Q_Perf**	Association (Composite Agg.)	New Property
34			memberEnd.type = Performance
35			memberEnd.upper = 1

#	CSMM	MAPPING	DIFFERENCES
36	**Q_Sec**	Association (Composite Agg.)	New Property
37			memberEnd.type = Security
38			memberEnd.upper = 1
39	**Perf_L**	Association (Composite Agg.)	New Property
40			memberEnd.type = Latency
41			memberEnd.upper = 1
42	**Perf_T**	Association (Composite Agg.)	New Property
43			memberEnd.type = Throughput
44			memberEnd.upper = 1
45	**Sec_DI**	Association (Composite Agg.)	New Property
46			memberEnd.type = Data Integrity
47			memberEnd.upper = 1
48	**Sec_Conf**	Association (Composite Agg.)	New Property
49			memberEnd.type = Confidentially
50			memberEnd.upper = 1

6. Deployment Artifacts

While the previous chapters introduced BSM as well as CSM of the service description method, this chapter outlines the Deployment Artifacts (DA) layer. Figure 6.1 depicts an overview of DA as part of SDM4SE (cf. figure 3.1) as well as the corresponding method engineering artifacts.

The intention of this layer is to generate technical specifications for the aforementioned service properties that are part of CSM. Possible result documents and hence appropriate technical specifications include WSDL, UDDI, WSMO, OWL-S, and SA-WSDL, which will be explained in section 6.1. As shown in figure 6.1, DA ascribes two abstract *activities* in order to complete this layer: (1) generate artifacts and (2) enhance artifacts. The first activity describes the automatic generation of technical specifications, such as WSDL [27] and UDDI [96]. The latter activity indicates that generated artifacts need to be enhanced with further technical details. The DA layer furthermore specifies two different *roles* who are responsible for the aforementioned activities. MDA experts have experience and own knowledge in the domain of model-driven development that includes modeling as well as transformations between models. In terms of SDM4SE, they are responsible to conduct the automatical generation of artifacts using transformation tools and transformation mappings. IT architects enhance generated artifacts using development *tools*, e.g., WSDL and UDDI editors. The CSMN UML Profile is used for displaying CSM result documents such as the one shown in the figures 5.10–5.15. Development tools may include WSDL as well as UDDI editors but also programming languages and XML editors. Transformation mappings refer to technical implementations of abstract mappings between different meta models.

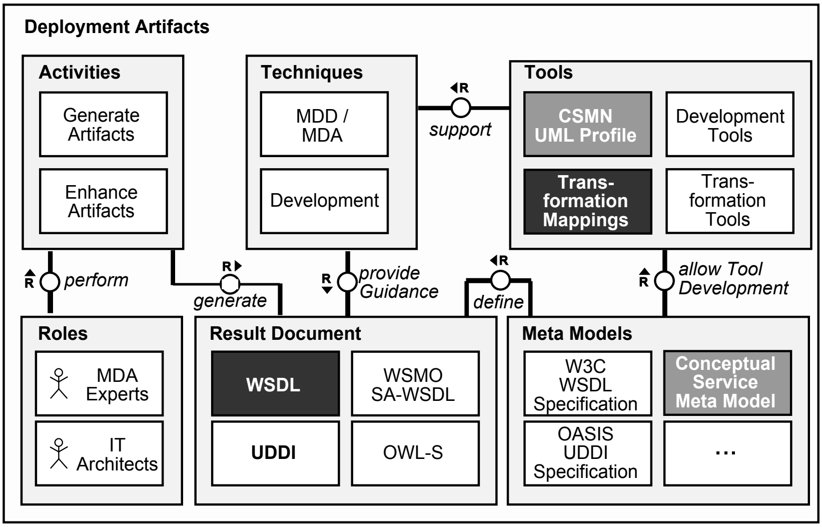


Figure 6.1.: DA Overview.

Transformation tools use these mappings in order to automatically generate deployment artifacts.

The black-shaded method engineering artifacts indicate the focus of this chapter. While section 6.1 analyses available technical specifications for service descriptions, section 6.2 introduces abstract mappings between CSMM and WSDL. This section also shows result documents for the Eco Value Calculation service.

6.1. Technical Specification Analysis

Scheithauer et al. [118] find the following specifications suitable for describing services in a technical manner: WSMO, OWL-S, SA-WSDL, WSDL as well as UDDI. While the latter two are well-know specifications for service-oriented architectures, WSMO, OWL-S, and SA-WSDL belong to the new field of *semantic web service* technology that relies on shared knowledge for improving describing and discovering services. The structure of this section is as follows: Subsection 6.1.1 presents the three specifications for semantic web services. While subsection 6.1.2 elaborates on WSDL, subsection 6.1.3 presents UDDI.

6.1.1. Semantic Web Services

The Web Service Modeling Ontology (WSMO) [112] is an ontology dedicated to web services. Its main elements include: (1) ontologies, (2) web services, (3) goals, and (4) mediators. The ontology element codifies domain knowledge which is used by all other elements. Web services' capabilities are expressed with pre- and postconditions to describe services' value offering. On the other hand, goal elements formalize desired value. Finally, mediator elements are means to overcome interoperability problems between other WSMO elements.

The Semantic Markup for Web Services (OWL-S) [74] is an upper ontology and based on the Web Ontology Language (OWL). It aims to describe web services in a semantic manner to enable automatic web service discovery, automatic web service invocation, and automatic web service composition and interoperation. OWL-S defines four main elements. The service element is the root element. The service profile element represents a service's functionality. The service grounding element discloses how to access the service. This is a bridge to a WSDL document. Finally, a service model element describes how a service works in terms of parameters and process descriptions.

Semantic Annotations for WSDL and XML Schema (SA-WSDL) [37] offers a way to annotate WSDL documents with semantic annotations. Whereas OWL-S brings its own means of grounding, WSMO uses SA-WSDL to do so.

6.1.2. Web Service Description Language (WSDL) [27]

WSDL is an industry-wide accepted standard from the W3C as well as a target platform for service-oriented architectures. The standard focuses on describing web services using XML with a functional character. The description includes message types, message formats, port types, operations, and protocol bindings. Listing A.3 outlines the complete WSDL meta model in the Ecore format and listing A.7 depicts the basic structure of WSDL documents.

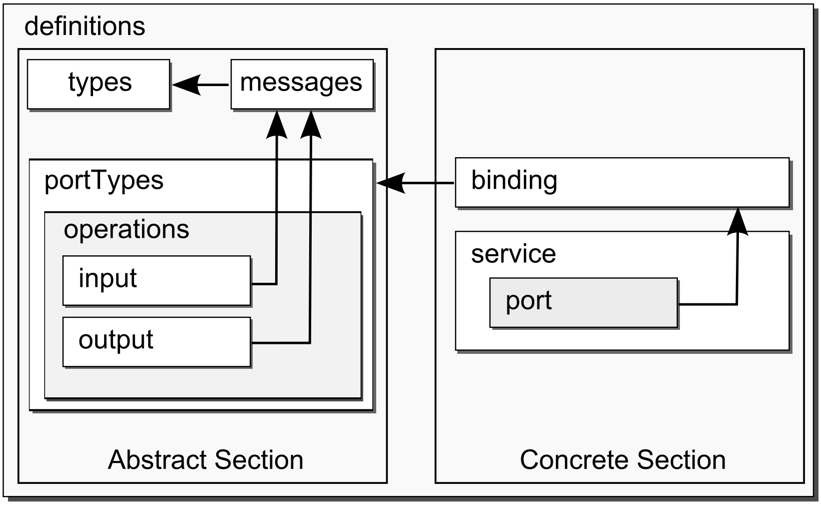


Figure 6.2.: WSDL Concepts.

The WSDL specification depicts six elements for defining services [27] that figure 6.2 shows. The *types* element allows specifying data types similar to the XML Schema Definition (XSD) [39]. The second element is *messages*, which are defined using the types element. The *messages* element define message formats that serve as input and output for operations. Lists of services' operations belong into the *portType* element. Whereas the elements *types, messages*, and *portType* belong to the *abstract section*, the elements *binding* and *service* belong to the concrete section. The difference lies in that the binding and service elements define concrete protocols and URIs, such as HTTP and SOAP.

6.1.3. The Universal Description, Discovery and Integration (UDDI) [96]

UDDI [96] is an OASIS standard for describing web services meta information in a service-oriented architecture. One concept of UDDI is to distinguish between different views on services' meta information. Whereas *white pages* list name and contact information for each service, *yellow pages* show a schema for service classifications.

6.2. Mapping between CSMM and WSDL

Following the introduction of CSMM as well as WSDL, this section develops an abstract mapping between the meta model and the technical specification in order to show their relationship as well as to provide a means for generating WSDL from any CSM diagram automatically. It is abstract in that it merely shows possible relationships between CSM and WSDL. Section 7.3, however, presents an implementation of this abstract mapping with model transformation technology. The abstract mapping consists of 32 mapping rules as shown below. The main objective is to map as much elements as possible. The rules, however, focus on WSDL's *abstract* elements types, portType, and messages. This is because CSM neither covers information about concrete bindings nor URIs. IT architects rather define these elements in the second activity *enhance artifacts*.

6.2.1. WSDL Definition Element

Mapping rules 6.1–6.18 concentrate on WSDL's *definitions* element. Rule 6.1 ascribes that the name attribute of the service product equals WSDL's definitions attribute name. Next to the name attribute, definitions may define *one* documentation. Mapping rules 6.2–6.18 specify how to generate the content for this element. Rule 6.2 for example tells that a concatenation of the string `key:` and the *key* attribute equal the *documentation* element. Another example is rule 6.6 that defines *for all* specified written languages a concatenation of the string `wr lang.:` and the *written language* attribute equal the *documentation* element.

Listing 6.1 shows the WSDL result document for the Eco Value Calculation service after applying rules 6.1 and 6.2. The rules are read as follows: whereas the left part is the source model, the right part is the target model. The source model is CSM and the target model WSDL. Each rule maps a source element to a target element.

$$ServiceProduct.name => wsdl : definitions.name \tag{6.1}$$

$$``key :'' + ServiceProduct.key => wsdl : documentation \tag{6.2}$$

$$``desc. :'' + ServiceProduct.description => wsdl : documentation \tag{6.3}$$

$$``doc. :'' + ServiceProduct.documentation => wsdl : documentation \tag{6.4}$$

$$\begin{gathered}\forall csmm.ServiceProduct.spokenlanguage \rightarrow ``spk\ lang. :'' \\ + ServiceProduct.spokenlanguage => wsdl : documentation\end{gathered} \tag{6.5}$$

$$\begin{gathered}\forall csmm.ServiceProduct.writtenlanguage \rightarrow ``wr\ lang. :'' \\ + ServiceProduct.writtenlanguage => wsdl : documentation\end{gathered} \tag{6.6}$$

$$``version :'' + ServiceProduct.version => wsdl : documentation \tag{6.7}$$

$$``created :'' + ServiceProduct.created => wsdl : documentation \tag{6.8}$$

$$``updated :'' + ServiceProduct.updated => wsdl : documentation \tag{6.9}$$

$$``nextupd. :'' + ServiceProduct.nextupdate => wsdl : documentation \tag{6.10}$$

$$``type :'' + ServiceProduct.type => wsdl : documentation \tag{6.11}$$

$$``automation :'' + ServiceProduct.automation => wsdl : documentation \tag{6.12}$$

$$``comp. :'' + ServiceProduct.composition => wsdl : documentation \tag{6.13}$$

$$``custom. :'' + ServiceProduct.customizable => wsdl : documentation \tag{6.14}$$

Listing 6.1: Generated WSDL with Mapping Rules 6.1 & 6.2

```
<wsdl:definitions name="Eco_Value_Calculation">
    <wsdl:documentation>
        key: SVC_KEY_ECOCALC_7493
    </wsdl:documentation>
</wsdl:definitions>
```

$$\text{“}paym.cond.:\text{''} + TermsofUse.paymentcondition => wsdl : documentation \tag{6.15}$$

$$\text{“}deli.cond.:\text{''} + TermsofUse.deliverycondition => wsdl : documentation \tag{6.16}$$

$$\forall Classification \rightarrow \text{“}class.:\text{''} + Classification.value \tag{6.17}$$
$$+\text{“}:\text{''}\ Classification.system => wsdl : documentation$$

$$\forall Benefit \rightarrow \text{“}benefit:\text{''} + Benefit.name => wsdl : documentation \tag{6.18}$$

6.2.2. WSDL Type Element

The mapping rules 6.19–6.22 focuses on generating the WSDL *type* element. Rule 6.19 specifies that for each resource an accordant XSD element must be created and named after the resource itself. The resources' other attributes *description* and *types* find no direct corresponding attribute in the XSD specifications. Hence rules 6.20 and 6.22 map these to the *documentation* attribute. Listing 6.2 shows the generated WSDL excerpt for rules 19–21.

$$Resource.name => xsd : element.name \tag{6.19}$$

$$\text{“}Resource:\text{''} + Resource.description => \tag{6.20}$$

$$xsd : element.annotation[1].documentation \tag{6.21}$$

$$\text{“}Type:\text{''} + Resource.type => xsd : element.annotation[1].documentation \tag{6.22}$$

Listing 6.2: Generated WSDL with Mapping Rules 6.19–6.22

```
...
<xsd:schema targetNamespace="http://62.52.175.245
    :8080/texo/">
    <xsd:element name="Certificate">
        <xsd:annotation>
            <xsd:documentation>
                description: Certifies an
                    appropriate Carbon Dioxide
                    Level
            </xsd:documentation>
            <xsd:documentation>
                type: Physical Good
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence />
        </xsd:complexType>
    </xsd:element>
    ...
</xsd:schema>
...
```

6.2.3. WSDL Messages Element

Mapping rules 6.23–6.28 focus on the generation of the *messages* element. Listing 6.3 shows the generated WSDL excerpt for all *message* related mapping rules. The strategy for generating messages is that for each capability exactly *one* input and *one* output message is created. Pre and post conditions with their corresponding resources map to WSDL's *part* element that is owned by the *messages* element. So all WSDL:parts that are pre condition elements belong to the one input message and consequently, all WSDL:parts that are post condition elements group in the output message.

$$Capability.name + “Input'' => wsdl : message[1].name \tag{6.23}$$
$$Capability.name + “Output'' => wsdl : message[2].name \tag{6.24}$$
$$Precondition.name => wsdl : message[1].part.name \tag{6.25}$$
$$Precondition.Con\_R.name => wsdl : message[1].part.element \tag{6.26}$$
$$Postcondition.name => wsdl : message[2].part.name \tag{6.27}$$
$$Postcondition.Con\_R.name => wsdl : message[2].part.element \tag{6.28}$$

Rules 6.23 and 6.24 specify for each service capability property a WSDL:message and map the capability name attribute to the name attribute of the messages. Listing 6.3 outlines in line 2 the input message for the capability *Issue Carbon Dioxide Certificate* and in line 8 the corresponding output message.

Following the messages element creation, rules 6.25 and 6.26 specify how to translate pre conditions into WSDL:part elements for input messages. Rule 6.25 tells that for each pre condition of a capability the pre condition's name attribute is mapped to a new WSDL part's name attribute. Rule 6.26 on the other hand, specifies the corresponding resource, which the pre condition references using the `Con_R` relationship, that maps to the WSDL part's element attribute. This element name matches the name of the XSD:element element of rule 6.19. Line 3 of listing 6.3 shows the part for the input message of the capability *Issue Carbon Dioxide Certificate*. The name attribute equals the pre condition's name and the element attribute matches the corresponding resource *Eco Value* (cf. figure 5.13).

Listing 6.3: Generated WSDL with Mapping Rules 6.23–6.28

```
...
<wsdl:message name="
     Issue_Carbon_Dioxide_CertificateInput">
    <wsdl:part element="tns:Eco_Value" name="
         Calculated_Eco_Value" />
</wsdl:message>
<wsdl:message name="Calculate_Carbon_DioxideInput">
    <wsdl:part element="tns:Material" name="
         Available_Material" />
</wsdl:message>
<wsdl:message name="Calculate_Carbon_DioxideOutput"
     >
    <wsdl:part element="tns:Eco_Value" name="
         Calculated_Eco_Value" />
</wsdl:message>
<wsdl:message name="
     Issue_Carbon_Dioxide_CertificateOutput">
    <wsdl:part element="tns:Certificate" name="
         Issued_Certificate" />
</wsdl:message>
```

6.2.4. WSDL portType Element

Listing 6.4 shows the generated WSDL excerpt for all *portType* related mapping rules. Rule 6.29 ascribes to name the *portType* with the name of the service product as shown in listing 6.4 in line 2. The mapping rules 6.30–6.33 apply for each capability. Rule 6.30 maps capabilities' names to WSDL operation names (cf. line 3 & 10). Rule 6.30 situates capabilities' *description* attribute to the WSDL operation attribute *documentation*. Rules 6.32 as well as 6.33 are used to include the input and the output messages that were previous mentioned with the mapping rules 6.23–6.28.

$$ServiceProduct.name => wsdl : portType.name \tag{6.29}$$

$$Capability.name => wsdl : operation.name \tag{6.30}$$

$$Capability.description => wsdl : operation.documentation \tag{6.31}$$

$$Capability.name + “Input'' => wsdl : operation.input.message \tag{6.32}$$

$$Capability.name + “Output'' => wsdl : operation.output.message \tag{6.33}$$

Listing 6.4: Generated WSDL with Mapping Rules 6.29–6.33

```
...
<wsdl:portType name="Eco_Value_Calculation">
    <wsdl:operation name="Calculate_Carbon_Dioxide">
        <wsdl:documentation>
            Calculates the carbon dioxide level for a
                given material.
        </wsdl:documentation>
        <wsdl:input message="
            tns:Calculate_Carbon_DioxideInput" />
        <wsdl:output message="
            tns:Calculate_Carbon_DioxideOutput" />
    </wsdl:operation>
    <wsdl:operation name="
        Issue_Carbon_Dioxide_Certificate">
        <wsdl:documentation>
            Issues a certificate that states an
                approprate carbon dioxide level for a
                 given material.
        </wsdl:documentation>
        <wsdl:input message="
            tns:Issue_Carbon_Dioxide_CertificateInput
            " />
        <wsdl:output message="
            tns:Issue_Carbon_Dioxide_CertificateOutput
            " />
    </wsdl:operation>
</wsdl:portType>
...
```

Part III.

Implementation and Evaluation

7. Tools and Transformations

After the presentation of SDM4SE in the four previous chapters, this and the following chapters implement and evaluate the method. Whereas this chapter presents the SDM Toolset, a complete technical implementation for the method, chapter 8 applies the method and the tool set in two different domains in order to show their performance in real-world settings.

The *SDM Toolset* supports modeling experts throughout the service description development process. It bundles a set of generic tools and two tools which are specific to service description modeling. The SDM Toolset is an implementation of the service description method and its artifacts. In particular, it allows modeling of BSM and CSM result diagrams, and the transformation and modifications of WSDL files. The tool set utilizes a generic architecture for model-driven development [51] that offers a means for modeling, transformation, and development of domain-specific languages. This platform made it possible to develop and to integrate modeling tools for the service description method as well as to implement the abstract mappings between CSMM and WSDL for automatic WSDL generation. The overall procedure was to setup appropriate tools for the generic architecture firstly, secondly to build the specific tools for BSMN and CSMN, and thirdly to develop the transformation between CSMM and WSDL.

The remaining structure of the chapter is that while section 7.1 introduces the generic architecture for model-driven development, section 7.2 presents the development of tools for BSM and CSM, respectively. Section 7.3, finally, depicts the creation of the CSMM-WSDL-transformation script.

7.1. Generic Architecture for Model-Driven Development

This section elaborates on the generic architecture for model-driven development. *Model-driven Development* (MDD) [81] is a well accepted software design approach that intends to improve software projects with respect to portability, productivity, and quality. In general, software architects gather business requirements in a specification document that software developers implement, which is impeded by the different mind sets of software architects and software developers, the time that the requirement implementation takes, and the inability to react to rapid requirement changes [125]. In contrast, MDD prescribes the utilization of models in order to specify requirements as well as to generate valid software artifacts automatically from these models. One approach for MDD is OMG's *Model-driven Architecture* (MDA) [81]. OMG specifies a set of other specifications around MDA that the following paragraphs detail along the generic architecture for MDD.

Hu and Scheithauer [51] describe a generic architecture for MDD, which figure 7.1 depicts. The intention of this architecture is to provide an overview of components for implementing MDA for any problem domain as well as offering a deciding foundation for choosing appropriate tools for each MDA component. Moreover, Hu and Scheithauer offer an analysis of currently available tools for realizing MDD that table 7.1 shows. This architecture differentiates between three major technologies: (1) Domain-specific Language (DSL) Development Technology, (2) Modeling Technology, and (3) Transformation Technology.

DSL Development Technology enables DSL experts to build domain-specific graphical and textual languages. The *Meta Modeling Framework* component allows building meta models for specific domains (*abstract language meta models*). OMG specifies the Meta-Object Facility (MOF) [84] standard for meta modeling and domain-specific language definition. For example the formal model of UML is represented in MOF. The Eclipse

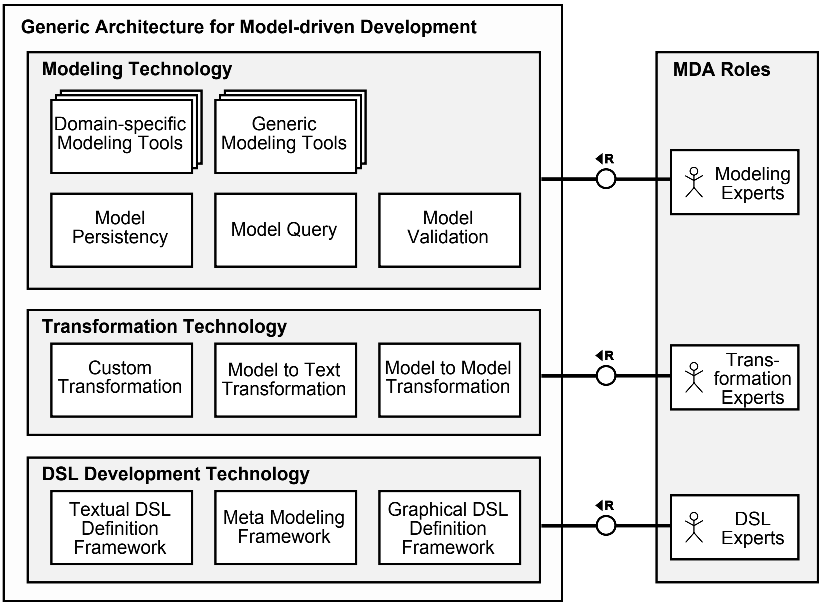

Generic Architecture for Model-driven Development
Modeling Technology
Domain-specific Modeling Tools
Generic Modeling Tools
Model Persistency
Model Query
Model Validation
Transformation Technology
Custom Transformation
Model to Text Transformation
Model to Model Transformation
DSL Development Technology
Textual DSL Definition Framework
Meta Modeling Framework
Graphical DSL Definition Framework
MDA Roles
R
Modeling Experts
R
Trans-formation Experts
R
DSL Experts

Figure 7.1.: Generic Architecture for Model-driven Development (cf. [51])

Modeling Framework (EMF) is an appropriate tool for this component for it realizes OMG's MOF specification. The *Graphical* and the *textual DSL Definition Framework*, on the other hand, enable the adaptation of abstract language meta models into *concrete language meta models*. These concrete language meta models define next to the domain how languages *look* and *feel* for modeling experts. This includes, for example, how to graphically represent domain-specific elements and their relationships. Available tools for defining concrete DSLs are Eclipse's Graphical Modeling Framework (GMF) and Eclipse's Textual Modeling Framework (TMF). It is important to note that TMF was formerly part of the openArchitectureWare (oAW) project.

Modeling Technology is the toolkit for modeling experts. *Domain-specific Modeling Tools* refer to tools that are specific to a domain. In case of

Table 7.1.: Appropriate Tools for the Generic MDD Architecture.

DSL Development Technology	
Textual DSL Definition FW	Eclipse Textual Modeling Framework (TMF)
Meta Modeling Framework	Eclipse Modeling Framework (EMF)
Graphical DSL Definition FW	Eclipse Graphical Modeling Framework (GMF)
Modeling Technology	
Domain-specific Modeling Tools	Business Service Modeling Tool
	Conceptual Service Modeling Tool
Generic Modeling Tools	UML Modeling Tool
	BPMN Modeling Tool
	SBVR Modeling Tool
	WSDL Tool
	...
Model Persistency	XML Metadata Interchange (XMI) [86]
Model Query	Object Constraint Language (OCL) [82]
Model Validation	Object Constraint Language (OCL) [82]
Transformation Technology	
Custom Transformation	Black Box for QVT [89]
Model to Text Transformation	Acceleo
	Xpand
Model to Model Transformation	Procedural QVT [89]
	Declarative QVT [89]
	Atlas Transformation Language (ATL) [36]

SDM4SE, table 7.1 lists the Business and the Conceptual Service Modeling Tool. *Generic Modeling Tools* on the other hand, include general purpose modeling tools, such as Eclipse's Model Development Tools [1] that offers next to others UML, BPMN [93], and SBVR [90]. Generic tools furthermore may include editors for WSDL [27] and XSD [39]. Modeling technology also offers a means for *Model Persistency* in order to store diagrams as well as to interchange these between tools and actors. OMG prescribes the XML Metadata Interchange (XMI) [86] specification for doing so. Additionally, once modeling experts finish modeling, it is volitional to check diagrams against structural rules so that the diagrams conform to their meta models, and hence, to their domain. Modeling experts may use the *Model Validation* functionality of the architecture. Another feature is *Model Query* for accessing diagram data in a programmatic manner. OMG offers the Object Constraint Language (OCL) [82] for model validation and model querying.

Transformation Technology, finally, offers a means for transforming information from one language to another. Possible examples include the transformation of conceptual models into deployment artifacts such as BPMN [93] to BPEL [10] transformation [120] and UML Class Diagrams to XSD transformation [51]. As shown in figure 7.1, transformation technology distinguishes three categories. *Model to Model Transformation* is dedicated to transforming information between models. OMG specifies Query/Views/Transformations (QVT) [89] for this purpose, which relies on other OMG standards (e.g. OCL) in order to define queries and constraints on models during transformation. Another language for model to model transformation is the Atlas Transformation Language (ATL) [36]. *Model to Text Transformation*, on the other hand, addresses transformations between models and text artifacts, such as software code, reports, or documents. For this type of transformation, OMG depicts the MOF Models to Text Transformation Language (MOFM2T) [88]. Appropriate tools for model to text transformations include Xpand and Acceleo that are both part of Eclipse's Model to Text (M2T) project [2]. *Custom Transformation*, finally, refers to QVT's *black box implementation*, which foresees writing custom transformations in other languages (e.g. Java) for complex transformations that cannot be expressed in QVT itself.

Following the considerations about a generic MDD architecture, figure 7.2 depicts the concrete composition of appropriate tools for the SDM4-SE Toolset. The main objective was to find a rich platform for the SDM4SE implementation, which Eclipse and its sub projects provide. SDM4SE Toolset is subdivided into four major components. On the figure's right hand side is the *Eclipse Modeling Project* [3] that bundles next to others the following tools. EMF and GMF for DSL development, QVT and ATL

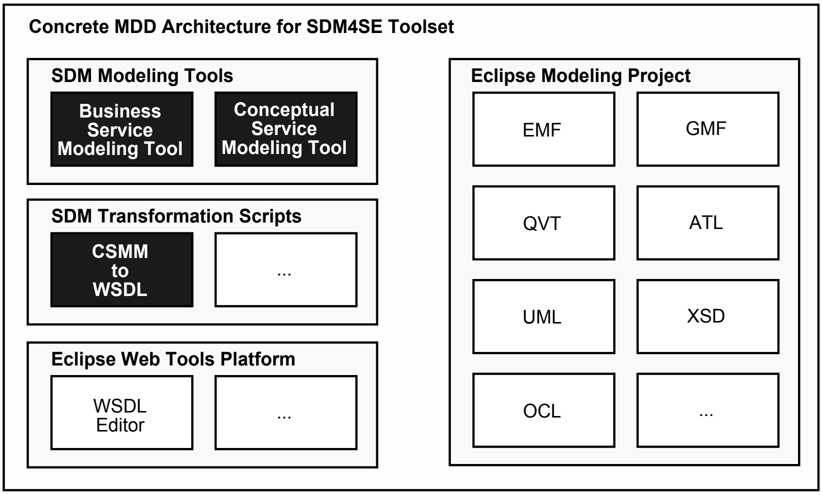

Concrete MDD Architecture for SDM4SE Toolset
SDM Modeling Tools
Business Service Modeling Tool
Conceptual Service Modeling Tool
SDM Transformation Scripts
CSMM to WSDL
...
Eclipse Web Tools Platform
WSDL Editor
...
Eclipse Modeling Project
EMF
GMF
QVT
ATL
UML
XSD
OCL
...

Figure 7.2.: Concrete MDD Architecture for the SDM4SE Toolset

for model to model transformation as well as UML and XSD for general purpose development. On the lower left hand side of the figure is the *Eclipse Web Tools Platform* that contributes a WSDL Editor. The remaining two components represent implementations for SDM4SE that have been designed and implemented during the project described here. The *SDM Modeling Tools* are shown in the figure's upper left. The component comprises two tools. The Business and the Conceptual Service Modeling Tool allow documenting BSM and CSM, respectively. The *SDM Transformation Scripts* component below shows one transformation script that implements the abstract mapping rules between CSMM and WSDL for automatic artifact generation.

7.2. Service Description Modeling Tools

This section elaborates on the development of the two tools *Business Service Modeling Tool (BSMT)* and *Conceptual Service Modeling Tool (CSMT)* using Eclipse technology (cf. DSL Development Technology in table 7.1). Figure 7.3 outlines how to develop graphical DSLs using Eclipse EMF and GMF technology [4].

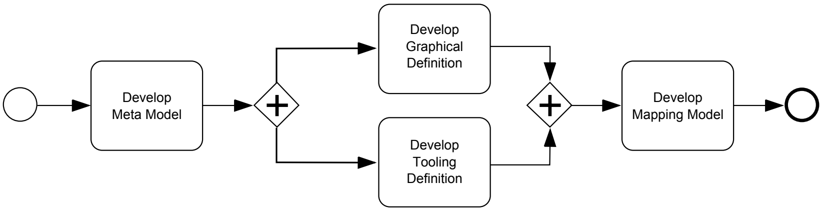


Figure 7.3.: Simplified DSL Development Process using Eclipse GMF & EMF (cf. [4])

EMF provides the technology in order to setup a meta model according to OMG's MOF standard. As aforementioned, meta models for designing languages define the *abstract syntax*. EMF provides the Ecore editor for specifying meta models. GMF, on the other hand, provides the technology that defines a *concrete syntax* for new graphical languages.

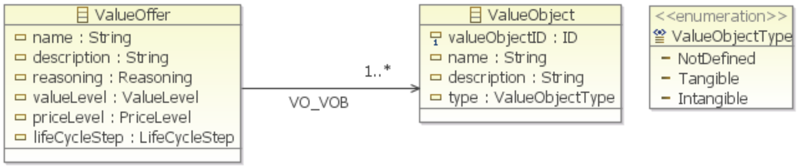
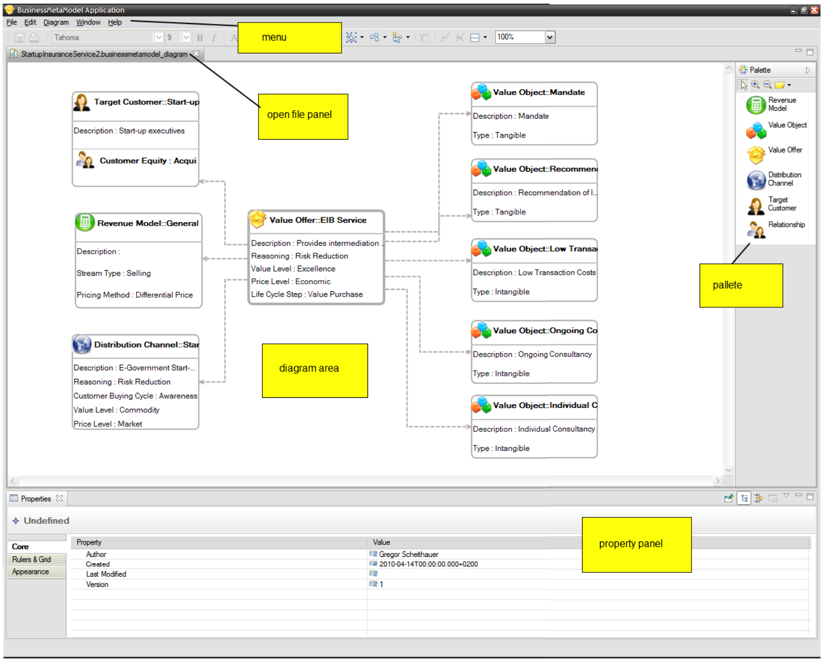


Figure 7.4.: Excerpt of BSMM in Ecore

The first step *Develop Meta Model* refers to establishing a meta model using EMF and the Ecore Editor. Figure 7.4 shows an excerpt of BSMM's Ecore representation. The Ecore element `EClass` represents the BSMM elements *Value Offer* and *Value Object* and the `EAttribute` BSMM's attributes. The *Value Object Type* enumeration is represented by Ecore's `EEnum` element. The *VO_VOB* relationship is represented by an `ERefer-`


BusinessMetaModel Application
File Edit Diagram Window Help
menu
StartupInsuranceService2.businessmetamodel_diagram
open file panel
Target Customer::Start-up
Description : Start-up executives
Customer Equity : Acqui
Revenue Model::General
Description :
Stream Type : Selling
Pricing Method : Differential Price
Value Offer::EIB Service
Description : Provides intermediation
Reasoning : Risk Reduction
Value Level : Excellence
Price Level : Economic
Life Cycle Step : Value Purchase
Distribution Channel::Star
Description : E-Government Start-...
Reasoning : Risk Reduction
Customer Buying Cycle : Awareness
Value Level : Commodity
Price Level : Market
diagram area
Value Object::Mandate
Description : Mandate
Type : Tangible
Value Object::Recommen
Description : Recommendation of l...
Type : Tangible
Value Object::Low Transa
Description : Low Transaction Costs
Type : Intangible
Value Object::Ongoing Co
Description : Ongoing Consultancy
Type : Intangible
Value Object::Individual C
Description : Individual Consultancy
Type : Intangible
Palette
Revenue Model
Value Object
Value Offer
Distribution Channel
Target Customer
Relationship
pallete
Properties
Undefined
Core
Rulers & Grid
Appearance
property panel

Property	Value
Author	Gregor Scheithauer
Created	2010-04-14T00:00:00.000+0200
Last Modified	
Version	1

Figure 7.5.: Business Service Modeling Tool Screenshot

ence element. Listings A.1 and A.2 show the complete Ecore models for BSMM and CSMM, respectively.

In the step *Develop Graphical Definition* the DSL expert decides which elements need a graphical representation. For each qualifying element, the expert defines graphical nodes. Graphical nodes include rectangles, ellipses, and circles. After finishing the graphical definition, the DSL expert creates palettes and menus for the diagram editor in the *Develop Tooling Definition* step so that modeling experts can use them in order to drag new elements into diagrams.

In the final step *Develop Mapping Model* the DSL expert combines the three artifacts; the *Meta Model*, the *Graphical Definition*, and the *Tooling Definition*. The resulting mapping model contains links between elements from the meta model and the graphical nodes that were created in the graphical definition step. Likewise, links must be created between meta model elements and pallets and menus.

Provided the DSL expert completed all steps, the GMF tool automatically generates a diagram editor. Figure 7.5 shows the modeling tool for BSM. The right hand side features a *palette* that shows the five elements that belong to BSMM. Users may use the palette in order to create new elements for their diagrams. The figure's bottom holds the *property* panel. This panel displays properties from selected elements. Currently no element in particular is selected, and hence, the panel displays the diagram's attributes that are author, created, last modified, and version. The tool also features a *menu* for file operations, editing and the like. Below is the *open file panel* that indicates currently open BSM diagrams. At the moment the tool shows merely one open file for a service called *Entrepreneur Insurance Bundle* which section 8.1 details. The *diagram area* in the middle of the figure allows creating and manipulating of diagrams.

Both, the Business Service Modeling Tool as well as the Conceptual Service Modeling Tool were developed using the DSL development process in figure 7.3. Screenshots of both tools are found in appendix B.

7.3. Service Description Transformation scripts

Whereas the previous section elaborated on modeling tools for BSM and CSM, this section presents a transformation implementation for the abstract mapping rules between CSMM and WSDL (cf. section 6.2). The listings 7.1— 7.7 show the implementation of these mapping rules using the transformation language *procedural QVT* of the Eclipse Modeling Project [3]. The complete procedural transformation script shows listing A.4.

However, the abstract mapping rules were also implemented with the aforementioned relational QVT and the ATL transformation languages.

Listings A.5 and A.6 depict these implementations for further reference.

These transformation rules merely cover the abstract section of WSDL elements that are *definitions*, *types*, *messages*, and *port types*. IT architects must complete the WSDL file in the *Enhance Artifacts* activity. This activity comprises the completion of types as well as to setup a concrete binding and a service endpoint.

Listing 7.1: Procedural QVT Transformation Script: Init

```
 1  import com.siemens.ct.texo.ise.m2m.qvt.oml.javaBlackbox.
        WSDLUtil;
 2  import com.siemens.ct.texo.ise.m2m.qvt.oml.javaBlackbox.XSDUtil
        ;
 3
 4  // Import of three Ecore meta models for the transformation
 5  modeltype CSM uses "http://www.siemens.com/CT/texo/ise/
        ConceptualServiceModel";
 6  modeltype WSDL uses "http://www.eclipse.org/wsdl/2003/WSDL";
 7  modeltype XSD uses "http://www.eclipse.org/xsd/2002/XSD";
 8
 9  // The transformation script
10  transformation CSM2WSDLTransform(in csm: CSM, out wsdl :WSDL);
11
12      // Constant initialization
13      helper targetNsPrefix()~: String = 'tns';
14      helper targetNs()~: String = 'http://www.example.org/
            NewWSDLFile/';
15
16      // Main procedure that is invoked on transformation start
17      main()~{
18          var wsdlDefinition:= csm.objectsOfType(ServiceProduct).
                map serviceProduct2wsdlDefinitions();
19      }
```

7.3.1. Transformation Initialization

Listing 7.1 shows the transformation script's beginning. Lines #1 and #2 depict an import of two utility classes for WSDL and XSD manipulation. These were implemented in the Java programming language and conform to OMG's QVT black box specification. These utility classes are necessary for retrieving qualified names from XML documents, a functionality that is not found in procedural QVT itself.

Ecore meta models that are necessary for the transformation are loaded in lines #5—#7. The *source* meta model is CSMM that was introduced in section 5.2. The *target* meta model is WSDL. However, in order to generate type elements in WSDL, it is crucial to load the XSD meta model as well.

Line #10 defines then the transformation `CSM2WSDLTransformation` with CSMM as input model and WSDL as output model. Line #17 defines the main procedure that is invoked for every transformation. It tells to map the `wsdlDefinition` element against the mapping `serviceProduct2wsdlDefinitions()`.

7.3.2. WSDL Definition Element

After the initialization of the transformation, the next listings show the implementation of abstract mapping rules. Listing 7.2 exemplary depicts the rules 1—14. Line #2 shows the main mapping `serviceProduct2wsdlDefinitions`. It is important to note the body of the mapping header in line #2:

```
mapping <context type>::<identifier»:«result parameters»
```

The `context type` refers to the context of the mapping that in this case is CSMM's Service Product element. While the `identifier` equals the mapping's name, the `result paramter` specifies the target element that in this case is WSDL's definition element. The two keywords `self` and `result` refer to the *context type* and to the result parameter, respectively.

The following lines #4—#8 setup the namespace for WSDL definitions element. Line #11 represents the implementation of mapping rule 1 that tells to map Service Products' names to WSDLs' definitions name attribute. The line also shows that before the *name* element is mapped, it executes an operation in order to replace space characters with an underline character, because the WSDL specification does not allow space characters for qualified names.

Lines #13 and #14 implement mapping rule 2. Firstly a WSDL documentation element is created and secondly the concatenation of the string `[key:`, the name of the key attribute, and the string `]` is mapped to the documentation element.

Listing 7.2: Procedural QVT Transformation Script: Mapping Rules 1—14

```
 1  // Main mapping
 2  mapping ServiceProduct::serviceProduct2wsdlDefinitions():wsdl::
         Definition{
 3      // Namespace setup up for WSDL definition element
 4      result.addNamespace('soap','http://schemas.xmlsoap.org/wsdl
            /soap/');
 5      result.addNamespace('wsdl','http://schemas.xmlsoap.org/wsdl
            /');
 6      result.addNamespace('xsd', 'http://www.w3.org/2001/
            XMLSchema');
 7      result.addNamespace(targetNsPrefix(), targetNs());
 8      result.targetNamespace :=  targetNs();
 9
10      // Mapping rule 1
11      result.qName := getQName(self.name.replace('␣','_'));
12      // Mapping rule 2
13      documentationElement := result.createDocumentation('wsdl');
14      documentationElement.appendTextContentToDOMElement('[key:'
            + self.key.repr()~+ ']');
15
16      // Mapping rules 3 & 4
17      ...
18
19      // Mapping rule 5
20      documentationElement.appendTextContentToDOMElement('[spkn␣l
            .:' +
21          self.spokenLanguage->iterate(l : Lang; langs : String =
                '' |  langs := langs + l.repr()~+ ';')~+ ']');
22
23      // Mapping rules 6 - 14
24      ...
25  };
```

Lines #20 and #21 depict a mapping between the spoken languages attribute to documentation element. Because it is possible to name more than one language for service products, the rule makes use of an iteration on all spoken languages.

Listing 7.3 shows the mapping for the *Terms of Use* element of CSMM's product category. Line #3 checks whether such an element was specified, and in the positive case, lines #5 and #6 store the element in the variable `tou`. Coherent with mapping rule 15, line #8, maps the concatenation of the string `[paym. cond.:`, the name of the payment condition attribute, and the string `]` to the documentation element.

Listing 7.3: Procedural QVT Transformation Script: Mapping Rules 15—16

```
 1  ...
 2      // in case the terms of use element was specfified
 3      if csm.objectsOfType(TermsOfUse)->size()~> 0 then {
 4          // Retrieve terms of use element
 5          var tou =
 6              csm.objectsOfType(TermsOfUse)->asOrderedSet()->
                    first();
 7          // Mapping rule 15
 8          documentationElement.appendTextContentToDOMElement('[
                paym.␣cond.:' + tou.paymentCondition.repr()~+ ']');
 9          // Mapping rule 16
10          documentationElement.appendTextContentToDOMElement('[
                deli.␣cond.:' + tou.deliveryCondition.repr()~+ ']')
                ;
11      }
12      endif;
13
14  ...
```

Listing 7.4 implements mapping rules 17 and 18 that ascribe to map all classification and benefit properties to WSDL's documentation element. Line #3 for example, tells that for each specified classification property the documentation element must be updated that is shown in line #5.

Listing 7.4: Procedural QVT Transformation Script: Mapping Rules 17—18

```
...
    // For each classification element
    csm.objectsOfType(Classification)->forEach(c){
        // Mapping rule 17
        documentationElement.appendTextContentToDOMElement('[
             classi.:' + c.value + '␣' + c.system.repr()~+ ']');
    };
    // For each benefit element
    csm.objectsOfType(Benefit)->forEach(b){
        // Mapping rule 18
        documentationElement.appendTextContentToDOMElement('[
             benefit:' + b.name + ']');
    };
...
```

7.3.3. WSDL Type Element

Listing 7.5 targets mapping rules 19—21 which define types for input and output messages. The mapping in line #1 tells the context type as *Resource* and the result parameters as *xsd::XSDElementDeclaration*. Line #3 implements mapping rule 19 in that the name attribute of an XSD type is mapped to the name attribute of the resource property without blank spaces. While line #5 creates a new annotation for the XSD type element, lines #8 and #10 map the resource description attribute as well as the resource type to the annotation element.

Listing 7.5: Procedural QVT Transformation Script: Mapping Rules 19—21

```
mapping Resource::resource2xsdElement(xsdSchema : XSDSchema): xsd::XSDElementDeclaration{
    // Mapping rule 19
    name := self.name.replace('␣','_');

    annotation := object XSDAnnotation{};

    // Mapping rule 20
    annotation.createUserInformation(xsdSchema, result, 'documentation', self.description);
    // Mapping rule 21
    annotation.createUserInformation(xsdSchema, result, 'resourcetype', self.type.repr());

    ...
}
```

7.3.4. WSDL Messages Element

Listing 7.6 considers the mapping rules 22—27 that address WSDL's messages element. The context type is CSMM's *Capability* and the result parameters the *message* element of WSDL. The mapping in line #1 builds the *input* message, and the mapping in line #17 builds the *output* message. Line #3 maps the concatenation of the capability name and the string `Input` to the message element's name attribute. Then in line #5 for all pre conditions of this capability which link to a resource, the name of the pre condition is mapped to the part element that belongs to the message element. Line #11 fills the elementName attribute of the part element with the name of the resource that is linked by the pre condition.

Listing 7.6: Procedural QVT Transformation Script: Mapping Rules 22—27

```
mapping Capability::capability2wsdlInputMessage()~: wsdl::Message{
    // Mapping rule 22
    result.qName := getQName(self.name.replace('␣','_')~+ 'Input');
    // For all pre conditions that link to a resource
    self.Capability_Condition->forEach(c | c.metaClassName()= 'PreCondition'
        and (not c.Condition_ref_Resource.oclIsUndefined())){
        result.eParts += object Part{
            // Mapping rule 24
            name := c.name.replace('␣','_');
            // Mapping rule 25
            elementName := getQName(targetNsPrefix()~+ ':' +
                                c.Condition_ref_Resource.name.replace('␣','_'));
        }
    };
}

mapping Capability::capability2wsdlOutputMessage()~: wsdl::Message{
    // Mapping rule 23
    result.qName := getQName(self.name.replace('␣','_')~+ 'Output');
    // For all post conditions that link to a resource
    self.Capability_Condition->forEach(c | c.metaClassName()= 'PostCondition'
        and (not c.Condition_ref_Resource.oclIsUndefined())){
        result.eParts += object Part{
            // Mapping rule 26
            name := c.name.replace('␣','_');
            // Mapping rule 27
            elementName := getQName(targetNsPrefix()~+ ':' +
                                c.Condition_ref_Resource.name.replace('␣','_'))
        }
    };
}
```

7.3.5. WSDL portType Element

This subsection finally presents the implementation of the mapping rules 28—32 that address the WSDL portType element. The portType represents the interface of the WSDL document in that it specifies operations and links operations with their input and output messages.

The mapping in line #1 of listing 7.7 specifies the *Service Product* as context type and the *wsdl portType* as result parameters. Mapping rule 28 in line #3 tells to map the service product's name to the name of the portType. Then for all capabilities, mapping rule 29 in line #9 ascribes to map the name of the current capability to the name of the WSDL operation. Line #16 specifies to map the capability's description attribute to the WSDL operation's documentation attribute.

Finally, lines #20 and #25 implement mapping rules 31 and 32, which map the aforesaid input and output messages to the corresponding operations.

Listing 7.7: Procedural QVT Transformation Script: Mapping Rules 28—32

```
mapping ServiceProduct::buildPortType()~: wsdl::PortType{
    // Mapping rule 28
    qName := getQName(self.name.replace('␣','_'));

    // For each Capability
    self.SProduct_Capability->forEach(c){
         eOperations += object Operation{
            // Mapping rule 29
            name := c.name.replace('␣','_');
            var inputMsgName := name + 'Input';
            var outputMsgName := name + 'Output';

            // Create documentation for operation
            documentationElement := createDOMElement('wsdl', '
                 documentation');
            // Mapping rule 30
            documentationElement.appendTextContentToDOMElement(
                 c.description);

            if (wsdl.objectsOfType(Message)->exists(qName.repr
                 ()~= inputMsgName))~then {
                // Mapping rule 31
                eInput.setMessageRef(targetNsPrefix()~+ ':' +
                     inputMsgName);
            }
            endif;
            if (wsdl.objectsOfType(Message)->exists(qName.repr
                 ()~= outputMsgName))~then {
                // Mapping rule 32
                eOutput.setMessageRef(targetNsPrefix()~+ ':' +
                     outputMsgName);
            }
            endif;
         }
    }
}
```

8. Case Studies

This chapter presents two case studies in real-world settings in contrast to the running example *Eco Calculator*. The first scenario taken from the insurance domain shows the development of the *Entrepreneur Insurance Bundle (EIB)* service that provides individual recommendations for insurances for entrepreneurs. The second scenario comes from an IT outsourcing scenario which depicts the development of the *Manage Client Hardware (MCH)* service that allows outsourcing hardware management.

The case studies' intention is to figure out whether the service description method and its artifacts support the *documentation*, *communication*, and *reasoning* of service descriptions on three different layers.

Data for both case studies were gathered using semi-structured interviews with subject-matter-experts and sifting through available documents that were related with the services. The author of this work was in the role of the *modeling expert* for the business and conceptual service model and in the roles of the *IT architect* and *MDA expert* for the deployment artifact layer. Interviewees, on the other hand, were in the roles of *business strategists* and *marketing experts* for the business service model and the conceptual service model, respectively. The *MCH* case study was performed in a greater setting. Scheithauer et al. [119] conducted a case study in the IT outsourcing domain for the service engineering approach *Integrated Service Engineering*. One part of the case study covered the service description method. In both cases it was necessary to modify the scenarios and to disguise company names for publication. The scenarios' scope and complexity remain the same, nevertheless.

Documentation and *transformation* were performed with the aforementioned *SDM Toolset*. BSM diagrams were developed with BSMT and CSM diagrams with CSMT. The resulting WSDL files were transformed using the QVTO transformation script.

The remaining chapter's structure is as follows. While section 8.1 presents the case study in the insurance domain, section 8.2 lays out the case study in the IT outsourcing domain. Each case study outlines a scenario description, and a walk-through of all three layers of the service description method that was presented in part II. Section 8.3 finally, discusses the findings of the case studies.

8.1. Case Study: Entrepreneur Insurance Bundle

This section presents a case study in the insurance domain. The first subsection introduces the scenario. The remaining subsections show SDM4-SE's application in order to implement a service description for BSM, CSM, and WSDL.

8.1.1. Scenario Description

The Chamber of Commerce and Industry (CCI) aims to improve the process of founding new companies by providing information about start-ups that includes information about business plans, finance plans, market analysis, marketing, insurance, and domain data [52]. In doing so, CCI foresees to reduce the time-to-market of new business ideas, create new jobs, increase wealth, and enlarge revenues.

For this, CCI setups an Entrepreneur Service Ecosystem. Figure 8.1 depicts a logical view on the Entrepreneur Startup Ecosystem, which follows the Aggregator type (cf. section 2.2). The middle layer shows that CCI acts as a platform provider, providing matching capabilities to entrepreneurs and service providers. Entrepreneurs may acquire access to it and search as well as select business services in order to support the business start-up

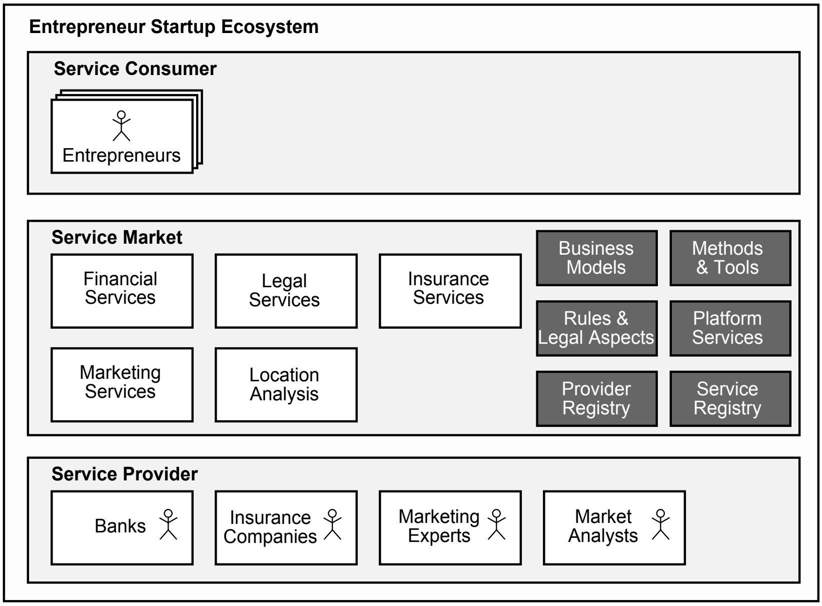


Figure 8.1.: Entrepreneur Startup Ecosystem

process. Any service provider – public or private sector – is encouraged to join the ecosystem.

An insurance broker wants to expand on the Entrepreneur Service Ecosystem, as a new delivery channel in order to reach new customers, and to generate revenue. The broker understands, that even though entrepreneurs know what insurance policies they need, they do not know the insurance market with its providers, nor the appropriate products. Table 8.1 shows recommended insurance policies.

Table 8.1.: Recommended Insurance Policies for Entrepreneurs, cf. [52]

Third Party Liability	Product Liability	Environmental Liability
Car Insurance	Interruption Insurance	Operation Cost Insurance
Legal Cost Insurance	Bad Dept Loss Insurance	Fire Insurance
Water Damage Insurance	Storm & Hail Insurance	

The broker develops a business idea: Entrepreneur Insurance Bundle Service. The idea is that the broker offers a recommendation for a complete and individualized insurance packet for entrepreneurs. Figure 8.2 shows a business model for the business idea utilizing the e^3 Value Ontology. It consists of the two market segments: Entrepreneurs and Insurance Providers, and the two actors: CCI and Insurance Broker.

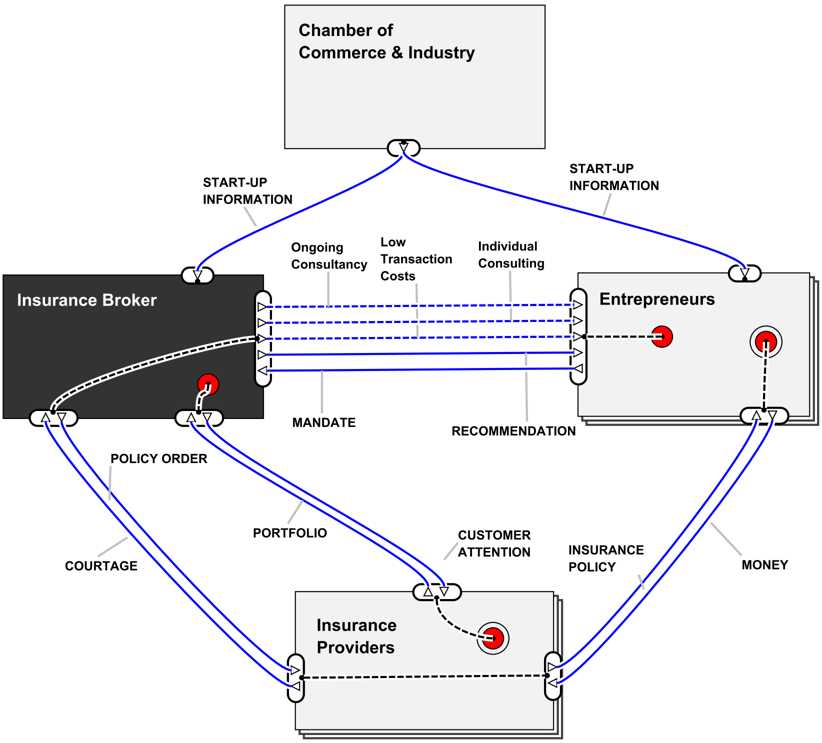


Figure 8.2.: Business Model for the Entrepreneur Insurance Bundle.

CCI (context provider & infrastructure provider (cf. section 2.2)) aims at improving the start-up process for entrepreneurs. The business model in figure 8.2 shows that CCI provides start-up information (Value Object) for both, insurance brokers and entrepreneurs. This information is valuable in that it comprises best-practices from numerous experiences in start-ups in the domains of business plans, finance plans, market analysis, marketing, and **insurance**. Entrepreneurs (service consumers) represent actors who are willing to start a new enterprise. For doing so, some juridical and recommended actions need to be performed. According to CCI [52], one recommended action is to acquire insurance policies. Insurance providers (service providers) provide insurance policies to companies as well as individuals for money. The insurance broker (context provider) on the other hand, acts as a single face to entrepreneurs and, in this case, matches a service to entrepreneurs' needs.

Entrepreneurs may utilize the start-up information to acquire insurance policies directly from insurance providers. Additionally, insurance providers provide their portfolio to insurance brokers in order to get more customer attention. The main value exchange, however, happens between entrepreneurs and the insurance broker, and between the insurance broker and insurance providers. The stimulus is that entrepreneurs rather acquire an insurance policy which matches their specific needs, than to buy general insurance policies from insurance providers directly. Exchanged tangible value objects between entrepreneurs & the insurance broker are mandates and recommendations. A mandate implies that an insurance broker may act in the name of entrepreneurs to acquire insurance policies. A recommendation implies a set of adapted insurance policies that matches entrepreneurs' individual needs. Next to these tangible values which are exchanged, three intangible values flow from the insurance broker toward entrepreneurs: low transaction costs, individual consulting, and ongoing consultancy. These values are so-called second-order-values [140] and are intangible and not actually transferred between actors. However, entrepreneurs gain these values additionally to the tangible value objects.

In the following, the service *Entrepreneur Insurance Bundle* from the view point of the insurance broker will be developed.

8.1.2. Business Service Model

The first activity includes defining a value offer. All following activities in this perspective take this outcome as a requirement document. Figure 8.3 shows the BSM result diagram that is detailed in the following paragraphs.

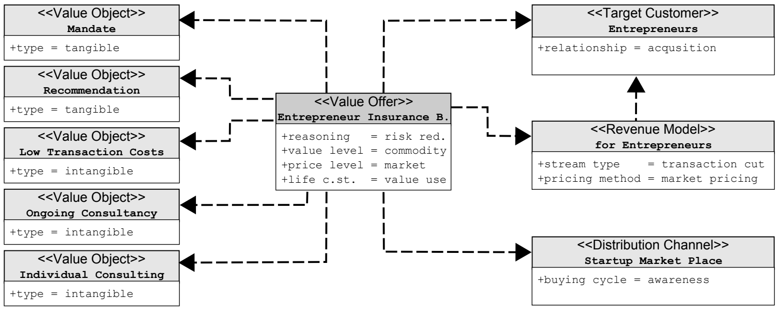


Figure 8.3.: Service Result Document

Entrepreneur Insurance Bundle (EIB) is the `Value Offer`. The reasoning is that it will reduce customers' risks in that insurance providers will compensate any insured losses. The value level is set to *commodity*, for the value offer is easy to imitate by competitors. The price level is situated as *market-based*. The value for customers are created while *value use* during the life cycle step.

Entrepreneurs are the service's `Target Customers`. The `Relationship` to these customers is not yet established. Hence the relationship is marked as *acquisition*.

The first tangible Value Object of the service is the *Mandate*, which is given to the insurance broker in order to perform the service. The second tangible value object is the *Recommendation* that contains a set of appropriate insurance policies. Outsourcing insurance policy acquirement to the insurance broker results in *lower transaction costs* for identification and contracting, *individual consulting*, and *ongoing consulting*, which are intangible value objects [140].

The *Entrepreneur Insurance Bundle* service's `Distribution Channel` relies solely on the CCI's service market place, which supports *awareness* state of customers' buying cycle.

Likewise, the insurance broker follows one `Revenue Model`, which settles for a *market-based price* since the insurance business is a high competitive one. Nevertheless, it is not considered as a commodity price for the recommendation's performance highly depends on insurance brokers' skills and knowledge about the market. The stream type is set to *transaction cut*, because brokers get a cut (courtage) of the money that is collected by insurance providers for insurance policies.

8.1.3. Conceptual Service Model

Following the completion of BSM, the next paragraphs outline the development of the result diagram for the CSM layer. For this, the modeling expert opens the Conceptual Service Modeling Tool (cf. section 7.2) and starts the new CSM diagram creation wizard. The general modeling procedure is to start with the product-related service properties, followed by the process properties. Following this, the modeling expert depicts people as well as physical evidence properties, before channels are set up and price properties are documented. Finally, the modeling expert establishes the promotion as well as the service's productivity and quality.

Figures 8.4–8.9 show the conceptual service diagram for the *Entrepreneur Insurance Bundle* service as the result documents for CSM, and hence, they show the application of CSMN. Listing A.12 shows the complete corresponding XML code for this diagram.

Establish Service Product Figure 8.4 depicts the product as well as the place and time category. The root of the conceptual service description is the **service product** property *Entrepreneur Insurance Bundle*. For identification, the key *EIB00001* is provided and further described as *Indi-*

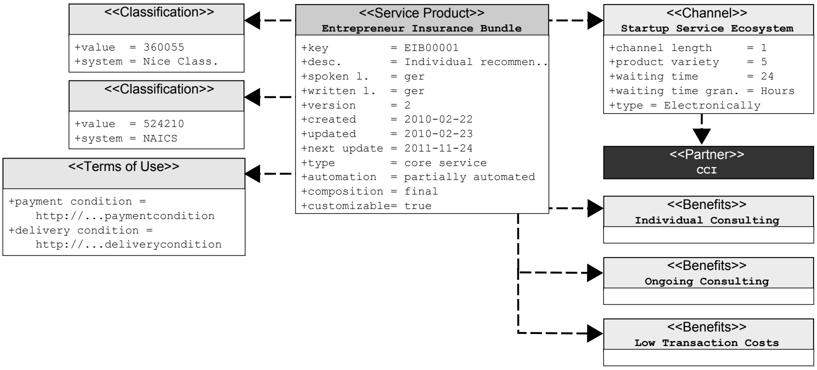


Figure 8.4.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: Product and Place & Time.

vidual recommendations for a complete and individualized insurance packet for entrepreneurs. The service is offered merely in German (*ger*). During the BSM modeling, entrepreneurs in generals were identified as the service's target customers (cf. section 8.1.2) influence decisions about which languages to offer. The EIB service version is *2*, was created on *2010-02-22*, last updated on *2010-02-23*, and will receive the next update at *2011-11-24*. From the perspective of the Insurance Broker, the EIB service is rather a *core service* as well as a *final service* and is *partially* automated, which is also influenced by identified target customers. It is important to note that potential customers may customize the service because customer-orientation is the differentiation for insurance brokers. The service features two **classifications**. The modeling expert documents the *Nice Classification* system with a value of *360055* that stands for *insurance consultancy*. The second classification utilizes the *NAICS* system with the value *524210* that relates to *insurance brokerages*. The **terms of use** for the EIB service points to *http://insurance-broker.com/paymentcondition* and *http://insurance-broker.com/deliverycondition*, respectively. The service product also lists three **benefits** that the modeling expert derives from the

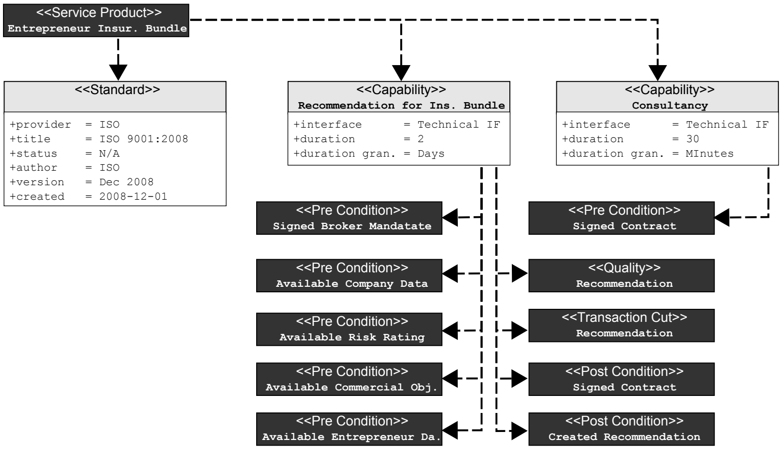


Figure 8.5.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: Process.

intangible value objects that were identified during the BSM layer. Benefits include *low transaction costs, individual consulting,* and *ongoing consulting.*

Setup Channels The conceptual service description defines the *Startup Service Ecosystem* as the **channel** which is derived from BSM's channel with a channel length of *1* since CCI mediates between the Insurance Broker and potential clients. Figure 8.4 depicts that the *electronically* channel overall features *5* products and has a maximal waiting time of *24 Hours*. Additionally, the channel property links to the partner property *CCI* who is the channel provider.

Define Service Process Figure 8.5 shows the part of the CSM diagram for properties in the process category for the EIB service. The modeling expert adds the two **capabilities** to the result diagram, which both were derived from the tangible value object *Recommendation*. The left hand

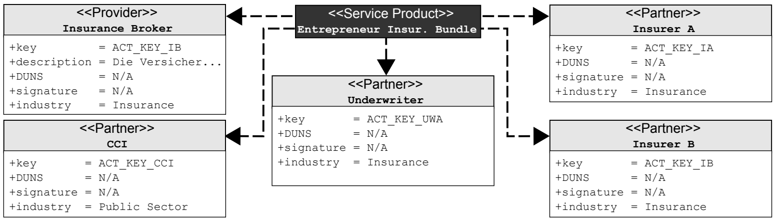


Figure 8.6.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: People.

side shows that the EIB service brings along the *ISO 9001:2008* **standard**, which indicates a quality management system of the Insurance Broker. The *Recommendation for Insurance Bundle* capability has a *technical interface* because most requests are made by phone, and with a duration of *2 Days*. Some requirements must be matched before the Insurance Broker is able to perform the recommendation for an insurance bundle. For the capability exist five pre conditions: a signed broker mandate, available information about company data, entrepreneur data, risk rating, and commercial object. The capability results in a signed contract and a created recommendation. Moreover, the capability refers to the transaction cut price *Recommendation* and the quality property *Recommendation* which both are discussed below. The second capability *Consultancy* refers to the ongoing and individual consultancy the Insurance Broker offers. It also features a *technical interface* with a duration of *24 Hours*. The single pre condition *signed contract* is required for this capability. All mentioned pre and post conditions are subject of the physical evidence part that one of the following paragraphs discusses (cf. figure 8.7).

Provide People Information After the modeling expert added the product, process, and channel properties to the diagram, the expert now documents information details about the service provider IT Company itself as

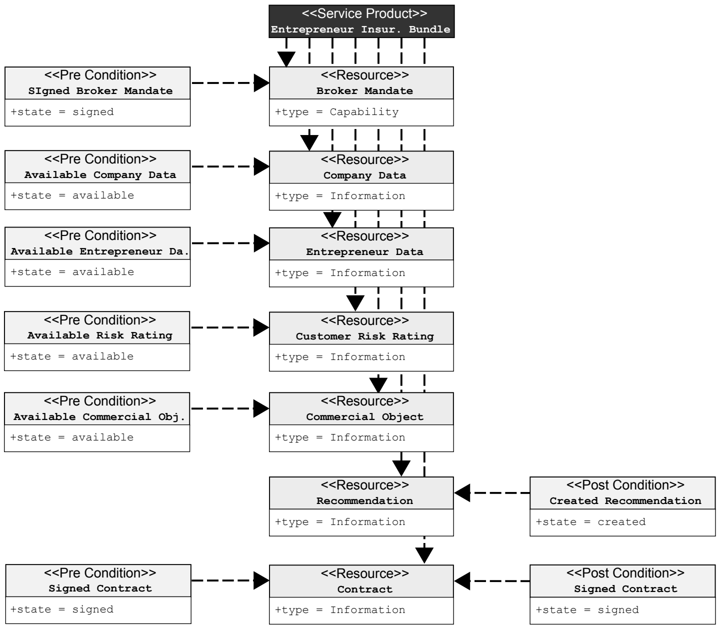


Figure 8.7.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: Physical Evidence.

well as its partners. Figure 8.6 shows that the service description defines five **actor** properties. The *Insurance Broker* is the service provider. The provider owns the unique key *ACT_KEY_IB* and is in the *Insurance* industry. Furthermore, four partners were identified for the broker. The two *Insurer* A and B provide the Insurance Broker with insurance products. The *Underwriter* is an unaffiliated partner who is associated with other insurers. Underwriters may sign contracts in the name of other insurers. The *CCI* partner, finally, is the chamber of commerce and industry that provides the market place for the Insurance Broker.

Determine Physical Evidence The modeling expert now models the service properties that belong to the physical evidence category. Figure 8.7

shows the identified resource and condition properties. The EIB service defines no less than seven **resources**. The resource *Broker Mandate* refers to the agreement between customers and Insurance Broker that the service provider may act in the name of the service customer, and hence falls in the category of *capability* resources. The remaining six resources classifies as *Information* resources. While *Company Data* is a set of information that is linked to the company, *Entrepreneur Data* is information about the entrepreneur. Also, entrepreneurs have different beliefs about risks and how much risks they want to cover with the insurance bundle. The *Customer Risk Rating* is a means to express this. The *Commercial Object* [1] represents the line of business. The *Recommendation* resource depicts the final set of insurances or individual insurance bundle for entrepreneurs and their companies. The *contract*, finally, is the agreement between customers and Insurance Broker for ongoing and individual consultancy.

Establish Pricing Figure 8.8 shows price and promotion properties for the EIB service. Firstly, the modeling expert reflects on service's target customer group and deduces the **payment** property *Bank Transfer* as the *preferred* way of payment with a *monthly* payment via *Bank Transfer*. The expert secondly set ups the **transaction cut** price *Recommendation*. The expert makes use of the target customer information but also finds valuable details in BSM's value offer's attributes *value level* and *price level* that ascribe the EIB service a *commodity* value level and an *market* price level. No concrete pricing is chosen because in the insurance industry brokers are not paid by their clients directly. Rather they *mediate* between new customers and insurance companies. Insurance brokers are then paid by insurance companies for each successful mediation. The capability *Recommendation for Insurance Bundle* links to this price.

[1] In German: *Unternehmensgegenstand*

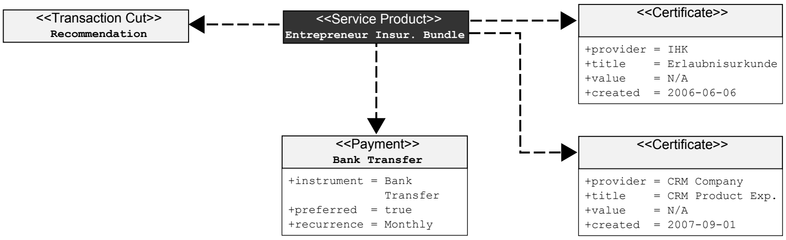


Figure 8.8.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: Price and Promotion.

Establish Promotion The right hand side of figure 8.8 shows two **certificate** promotion properties. The first one with the title *Erlaubnisurkunde*[2] is issued by the *IHK*[3] and allows conducting business in the insurance domain. The second one states that Insurance Broker is a *CRM Product Expert*.

Adjust Productivity and Quality The modeling expert finally documents productivity and quality properties that figure 8.9 presents. The EIB service defines the **quality** *Recommendation*, which is influenced by the targeted customer group and the service value level that was specified in the BSM result diagram. Moreover, the channel the service is provided over has an impact on the quality properties. The capability *Recommendation for Insurance Bundle* refers to this quality property. Hence an assertion can be made about the capabilities quality in terms of its dependability, performance, and security. The quality owns a **dependability** with an availability of *90.0%*. This is calculated with the proportion between the mean time to failure and mean time to repair (cf. subsection 5.2.8). Its reliability is set to *3 Days* as the mean time to failure and its maintainability to *8 Hours* as

[2] German for *concession*

[3] Chamber of Commercial and Industry

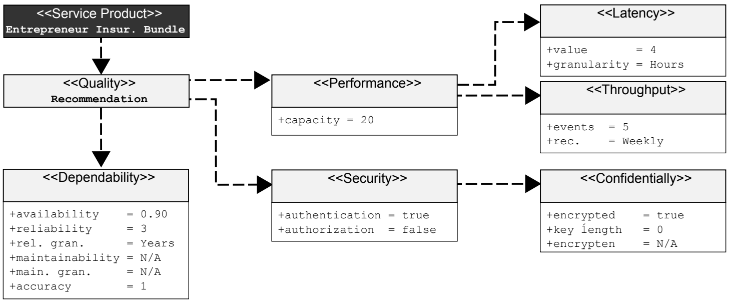


Figure 8.9.: Conceptual Service Diagram for *Entrepreneur Insurance Bundle* Service: Productivity & Quality.

the mean time to repair the service. The accuracy is set to *1* for no failure ever happened. Furthermore, the quality property outlines a **performance** with a capacity of *20* parallel requests, a **latency** of *4 Hours* and a **throughput** of *5 events Weekly*. It also depicts a **security** that provides *authentication* but no *authorization*. Moreover, the service's data is encrypted.

8.1.4. Deployment Artifacts

While the previous subsections showed the *Entrepreneur Insurance Bundle* service's result documents for BSM and CSM, this subsection discusses the generated WSDL file. The complete generated WSDL file is found in listing A.13. This DA's result document was automatically generated using the *Procedural QVT* transformation script introduced in section 7.3 (cf. also listing A.4). This WSDL file is not the final result document, however. It merely concludes the *Generate Artifacts* activity. IT architects must complete the WSDL file in the *Enhance Artifacts* activity. This activity comprises the completion of types as well as to setup a concrete binding and a service endpoint.

WSDL Definition Element Listing 8.1 shows the WSDL file's initial part. Line #2 outlines the WSDL definition element with the name attribute set to *Entrepreneur_Insurance_Bundle*. While lines #4—#13 show attributes of the service product property that were mapped to the documentation element, lines #14—#20 list information from the terms of use, classification, and benefit properties, such as the NAICS classification in line #17.

Listing 8.1: WSDL for *Entrepreneur Insurance Bundle* Service: Mapping Rules 1—18

```
 1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
 2  <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/
        soap/" xmlns:tns="http://www.example.org/NewWSDLFile/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="
        http://www.w3.org/2001/XMLSchema" name="
        Entrepreneur_Insurance_Bundle" targetNamespace="http://www.
        example.org/NewWSDLFile/">
 3      <wsdl:documentation>
 4          [key:EIB00001]
 5          [description:Individual recommendations for a complete
                and individualized insurance packet for
                entrepreneurs]
 6          [spkn l.:ger;][wr l.:ger;]
 7          [version:2]
 8          [created:Mon Feb 22 00:00:00 CET 2010]
 9          [updated:Tue Feb 23 00:00:00 CET 2010]
10          [next update:Thu Nov 24 00:00:00 CET 2011;]
11          [type:Core Service][automation:Partially Automated]
12          [composition:Final Service]
13          [customizable:true]
14          [paym. cond.:http://insurance-broker.com/
                paymentcondition]
15          [deli. cond.:http://insurance-broker.com/
                deliverycondition]
16          [classi.:360055 Nice Classification]
17          [classi.:524210 NAICS]
18          [benefit:Individual Consulting]
19          [benefit:Ongoing Consulting]
20          [benefit:Low Transaction Costs]
21      </wsdl:documentation>
```

WSDL Type Element Listing 8.2 outlines identified resources from the CSM diagram in figure 8.7. Lines #4, #10, #16, #22, #28, #34, and #40 show the corresponding XSD elements for each resource. In the following activity, IT architects need to refine these elements with concrete attributes.

Listing 8.2: WSDL for *Entrepreneur Insurance Bundle* Service: Mapping Rules 19—21

```
1  ...
2    <wsdl:types>
3      <xsd:schema targetNamespace="http://www.example.org/
           NewWSDLFile/">
4        <xsd:element name="Broker_Mandate">
5          <xsd:annotation/>
6          <xsd:complexType>
7            <xsd:sequence/>
8          </xsd:complexType>
9        </xsd:element>
10       <xsd:element name="Contract">
11         <xsd:annotation/>
12         <xsd:complexType>
13           <xsd:sequence/>
14         </xsd:complexType>
15       </xsd:element>
16       <xsd:element name="Recommendation">
17         <xsd:annotation/>
18         <xsd:complexType>
19           <xsd:sequence/>
20         </xsd:complexType>
21       </xsd:element>
22       <xsd:element name="Customer_Risk_Rating">
23         <xsd:annotation/>
24         <xsd:complexType>
25           <xsd:sequence/>
26         </xsd:complexType>
27       </xsd:element>
28       <xsd:element name="Company_Data">
29         <xsd:annotation/>
30         <xsd:complexType>
31           <xsd:sequence/>
32         </xsd:complexType>
33       </xsd:element>
34       <xsd:element name="Commercial_Object">
```

```
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Entrepreneur_Data">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
...
```

WSDL Messages Element Listing 8.3 presents the messages part of the transformation result. According to mapping rules 22—27 one input and one output message for each capability must be generated, provided the capability features at least one pre or post condition. Figure 8.5 outlines the two capabilities *Recommendation for Insurance Bundle* with five pre conditions and two post conditions, and *Consultancy* with just one pre condition. Line #2 shows the input message for the former capability. The message's name equals the capability's name plus the postfix *Input*. The capability's five pre conditions were transformed into message part elements as lines #3—#7 show. While the name attributes equal the pre conditions' names, the element references refer to linked resources. The output message is done likewise to the input message, with the difference that now post conditions were considered.

Listing 8.3: WSDL for *Entrepreneur Insurance Bundle* Service: Mapping Rules 22—27

```
...
  <wsdl:message name="Recommendation_for_Insurance_BundleInput">
    <wsdl:part element="tns:Broker_Mandate" name="Signed_Broker_Mandate"/>
    <wsdl:part element="tns:Company_Data" name="Available_Company_Data"/>
    <wsdl:part element="tns:Customer_Risk_Rating" name="Available_Risk_Rating"/>
    <wsdl:part element="tns:Commercial_Object" name="Available_Comm._Obj."/>
    <wsdl:part element="tns:Entrepreneur_Data" name="Available_Entrepreneur_Data"/>
  </wsdl:message>
  <wsdl:message name="ConsultancyInput">
    <wsdl:part element="tns:Contract" name="Signed_Contract"/>
  </wsdl:message>
  <wsdl:message name="Recommendation_for_Insurance_BundleOutput">
    <wsdl:part element="tns:Recommendation" name="Created_Recommendation"/>
    <wsdl:part element="tns:Contract" name="Signed_Contract"/>
  </wsdl:message>
...
```

WSDL portType Element Listing 8.4 presents the generated portType element that embodies information of CSM's capabilities. The portType element is named after the service product as line #2 tells. Lines #3 and #8 present WSDL operations for the service's capabilities. Both operations refer to the input messages that were presented in the previous paragraph. Because the *Consultancy* capability features no post conditions the operation features no output message, consequently.

Listing 8.4: WSDL for *Entrepreneur Insurance Bundle* Service: Mapping Rules 28—32

```
...
  <wsdl:portType name="Entrepreneur_Insurance_Bundle">
    <wsdl:operation name="Recommendation_for_Insurance_Bundle">
      <wsdl:documentation>Development of an individual
           recommendation for a set of insurance policies.</
           wsdl:documentation>
      <wsdl:input message="
           tns:Recommendation_for_Insurance_BundleInput"/>
      <wsdl:output message="
           tns:Recommendation_for_Insurance_BundleOutput"/>
    </wsdl:operation>
    <wsdl:operation name="Consultancy">
      <wsdl:documentation>Ongoing and individual consultancy.</
           wsdl:documentation>
      <wsdl:input message="tns:ConsultancyInput"/>
    </wsdl:operation>
  </wsdl:portType>
...
```

8.2. Case Study: Manage Client Hardware

After the insurance case study, this section presents a case study in the IT outsourcing domain. A real-world business service forms the basis for evaluating SDM4SE. A multi-national company offers a business service, namely the *Manage Client Hardware* service, that allows outsourcing the purchase and the maintenance of computer hardware like a desktop PC. The following subsections depict the case study's scenario, the implementation of the scenario for BSM and CSM, and finally conclude with a transformation into a WSDL document.

8.2.1. Scenario Description

IT Company is a multi-national firm that offers the business service *Manage Client Hardware*. The service's business model is to allow outsourcing of purchasing and the maintaining of computer hardware. Figure 8.10

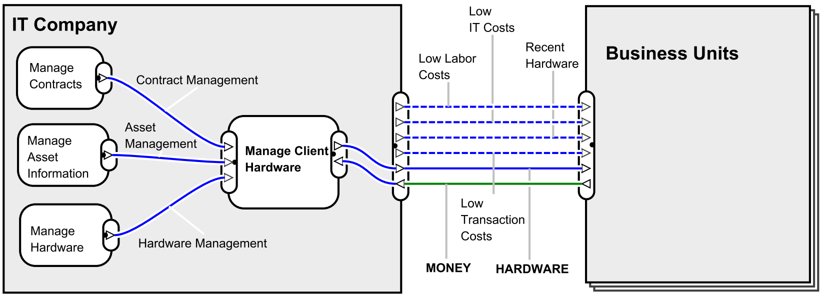
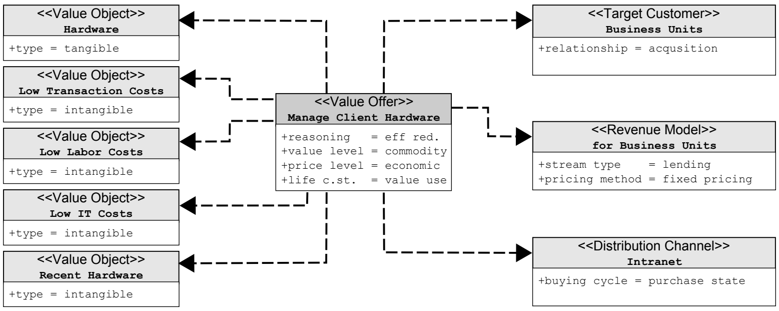


Figure 8.10.: Scenario End of a Leasing Contract

depicts the business model with the e^3 Value Ontology (cf. [43]). The business model comprises one actor with four value activities, a market segment, and nine value exchanges. The main actor is the IT Company itself. The company possesses three internal value activities: handle contracts, handle asset information, and handle hardware, with value exchanges toward the main value activity manage client hardware, which defines the external offered service. The market segment on the figure's right hand side pictures the company's target customers: its own business units. Between the actor and the market segment, the figure shows six value exchanges, and their corresponding value objects. The lowest one shows the value object *Money* that goes from the business units toward the IT Company. In this case, money is exchanged for the value object *Hardware*, which is directed from the company toward the business units. Next to these tangible values which are exchanged, four other values flow from the IT Company toward the Business Units: Low Transaction Costs, Low Labor Costs, Low IT costs, and Recent Hardware. These values are so-called second-order-values that are intangible and not actually transferred between the actors (cf. [140]). However, business units gain these values additionally to the main value objects. The three remaining value exchanges occur inside IT Company: Contract Management, Asset Management, and Hardware Management.

8.2.2. Business Service Model

The BSM layer transforms a service idea into a tangible foundation for business strategists, business analysts as well as business owners to decide whether to implement a service or not (predetermined breaking point).



<<Value Object>> Hardware
+type = tangible
<<Value Object>> Low Transaction Costs
+type = intangible
<<Value Object>> Low Labor Costs
+type = intangible
<<Value Object>> Low IT Costs
+type = intangible
<<Value Object>> Recent Hardware
+type = intangible
<<Value Offer>> Manage Client Hardware
+reasoning = eff red.
+value level = commodity
+price level = economic
+life c.st. = value use
<<Target Customer>> Business Units
+relationship = acqusition
<<Revenue Model>> for Business Units
+stream type = lending
+pricing method = fixed pricing
<<Distribution Channel>> Intranet
+buying cycle = purchase state

Figure 8.11.: Business Service Diagram: Manage Client Hardware.

The task includes eliciting and documenting knowledge about the *Manage Client Hardware* service for the following reasons: formalizing and communicating business ideas as well as to form a basis for service conceptualization and implementation. The scenario modeling follows the activities shown in figure 4.1: (1) establish value offer, (2) constitute value objects, (3) determine target customers, (4) determine relationship for each target customer, (5) determine distribution channel, and (6) setup appropriate revenue models.

Manage Client Hardware is the Value Offer. The reasoning is that it will reduce customers' effort in that the company will provide and maintain computer hardware. The value level is set to *commodity*, for the value offer is easy to imitate by competitors. The price level is situated as *economic*. The value for customers are created while *value use* during the life cycle step. The one tangible Value Object of the service is the *hardware object*. However, next to the hardware, there exist intangible value objects which also contribute to the service offering. Outsourcing hardware management to IT Company results in *lower transaction costs* for purchasing and contracting, *lower labor costs* for hardware maintenance, *lower IT costs,* and *state-of-the-art hardware. Business units* are the service's Target Customers. The Relationship to these customers is not yet established. Hence the relationship is marked as *acquisition.* The *Manage Client Hardware* service's Distribution Channel relies solely on the company's web online portal, which supports *purchase* state of customers' buying cycle. Likewise, the company follows one Revenue Model, which settles for a *fixed price* as pricing method and *lending* for the stream type.

Figure 8.11 depicts the final UML diagram (*resulting document,* cf. section 4) with the aforementioned Business Service Meta Model elements, which figure 4.2 prescribes. Additionally, listing A.14 shows the corresponding XML fragment for the UML diagram that can be used for persistence and further processing such as model transformation.

8.2.3. Conceptual Service Model

Following the completion of BSM, the next paragraphs outline the development of the result diagram for the CSM layer. For this, the modeling expert opens the Conceptual Service Modeling Tool (cf. section 7.2) and starts the new CSM diagram creation wizard. The general modeling procedure is to start with the product-related service properties, followed by the process properties. Following this, the modeling expert depicts people as well as physical evidence properties, before channels are setup and price properties are documented. Finally, the modeling expert establishes the promotion as well as the service's productivity and quality.

Figures 8.12—8.17 show the conceptual service diagram for the *Manage Client Hardware* service as the result documents for CSM and, hence, they show the application of CSMN. The whole diagram is divided into these six figures for practical reasons. Listing A.15 shows the complete corresponding XML code for the CSM result diagram.

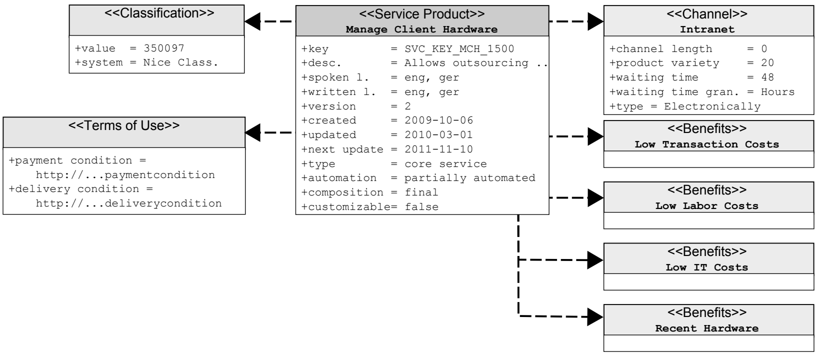


Figure 8.12.: Conceptual Service Diagram for *Manage Client Hardware* Service: Product and Place & Time.

Establish Service Product Figure 8.12 depicts the product as well as the place and time category. The root of the conceptual service description is the **service product** property *Manage Client Hardware*. For identification, the key *SVC_KEY_MCH_1500* is provided and further described as *allow outsourcing of purchasing and the maintaining of computer hardware*. The service is offered in two languages: *eng* and *ger*. Identified target customers (other business units (cf. section 8.2.2)) influence decisions about which languages to offer. The MCH service version is *2*, was created on *2009-10-06*, last updated on *2010-03-01*, and will receive the next update at *2010-11-10*. From the perspective of the IT Company, the MCH service is rather a *core service* as well as a *final service* and is *partially* automated, which is also influenced by identified target customers. Customers have no influence on the service's customization. The service features one **classification** using the *Nice Classification* system. Its value states *350097* which tells that the service belongs to the class *Outsourcing services [business assistance]*. The **terms of use** for the MCH service points to *http://itc.com/services/mch/paymentcondition* and

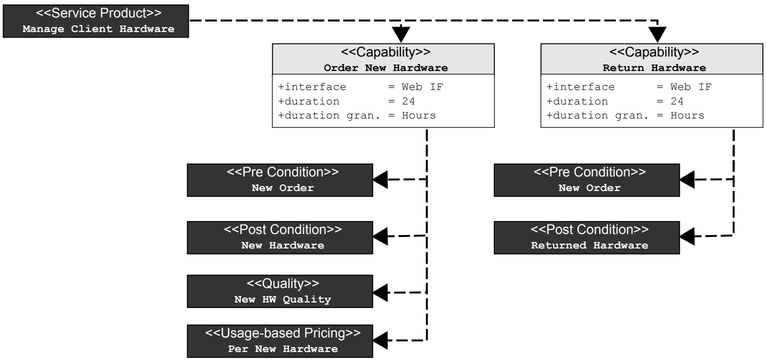


Figure 8.13.: Conceptual Service Diagram for *Manage Client Hardware* Service: Process.

http://itc.com/services/mch/deliverycondition, respectively. The service product also lists four **benefits** that the modeling expert derives from the intangible value objects that were identified during BSM. Benefits include *low transaction costs, low labor costs, low IT costs,* and *recent hardware.*

Setup Channels Finally, the conceptual service description defines the *intranet* as a **channel** which is derived from BSM's channel with a channel length of *0* since no partner is between the provider and possible consumers. The *electronically* channel overall features *20* products and has a maximal waiting time of *48 Hours.*

Define Service Process Figure 8.13 shows the part of the CSM diagram for properties in the process category for the MCH service. The modeling expert adds the two **capabilities** to the result diagram, which both were derived from the tangible value object *Hardware*. The *Order New Hardware* capability has a *web interface* and a duration of *24 Hours*. The capability owns the pre condition *New Order* and the post condition *New Hardware* which figure 8.15 shows in detail. Also, the capability links to the quality

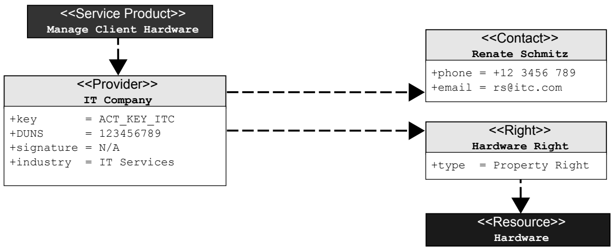


Figure 8.14.: Conceptual Service Diagram for *Manage Client Hardware* Service: People.

New HW Quality (cf. figure 8.17) and to the pricing *Per New Hardware* (cf. figure 8.16). The second capability *Return Hardware* also features a *web interface* with a duration of *24 Hours*. It owns the pre condition *New Order* and the post condition *Returned Hardware* (cf. figure 8.15).

Provide People Information After the modeling expert added the product, process, and channel properties to the diagram, the expert now documents information details about the service provider IT Company itself. Figure 8.14 shows that the service description defines merely one *actor*. The *IT Company* is the service provider. The provider owns the unique key *ACT_KEY_ITC* and is in the *IT services* industry. For the provider is the contact *Renate Schmitz* defined with the phone number *+12 3456 789* and the email *rs@itc.com*. Next to the contact information, IT Company holds the property **right** *Hardware Right* that links to the *Hardware* resource. This is done in order to attribute IT Company with the *ownership* of the hardware, which it lends to its customers.

Determine Physical Evidence Service properties that belong to the physical evidence category are shown in figure 8.15. The MCH service defines two **resources**. The *Hardware* is a *physical good* type resource. The *Order* resource on the other hand, is of type *information*. The modeling expert derives the *Hardware* resource from the *Hardware* value object of the BSM result diagram (cf. figure 8.11). Furthermore, three **conditions** exists, which were discussed along with the process properties (cf. figure 8.13). The modeling expert deduces these conditions from the aforementioned two capabilities. The pre condition *New Order* links to the *Order* resource and has the state *new*. It tells that a novel order must be created prior to order new hardware and prior to order to return hardware. The *New Hardware* post condition refers to the *Hardware* resource and depicts a state *new*. It means that new hardware is provided after accessing the *Order New Hardware* capability. The post condition *Returned Hardware* links as well to the *Hardware* resource and outlines its state as *returned*. This post condition indicates that after accessing the *Return Hardware* capability, the hardware is actually returned to IT Company.

Establish Pricing Figure 8.16 shows price and promotion properties for the MCH service. Firstly, the modeling expert reflects on the service's target customer group and deduces the **payment** property *Bank Transfer* as the *preferred* way of payment with a *monthly* payment via *Bank Transfer*. The second way of payment also uses the *bank transfer* but is due *yearly*. The expert secondly sets up the **usage-based pricing** *Per New Hardware* with an *exact* amount of *EUR 500* and an excluded tax of *0.19* percent. The expert makes use of the target customer information but also finds

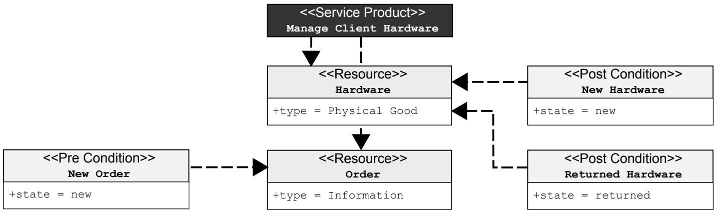


Figure 8.15.: Conceptual Service Diagram for *Manage Client Hardware* Service: Physical Evidence.

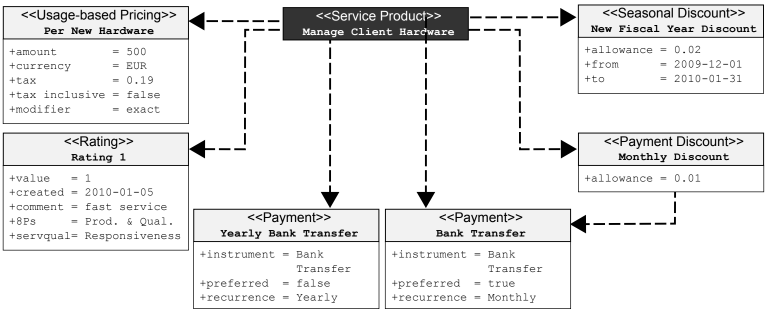


Figure 8.16.: Conceptual Service Diagram for *Manage Client Hardware* Service: Price and Promotion.

valuable details in BSM's value offer's attributes *value level* and *price level* that ascribe the MCH service a *commodity* value level and an *economic* price level. The capability *Order New Hardware* links to this pricing.

Establish Promotion The right hand side of figure 8.16 shows that the **Seasonal Discount** *New Fiscal Year Discount* gives an allowance of *2%* between the dates *2009-12-01* and *2010-01-31*. This discount should encourage business units to continue hardware ordering through the holiday season. The second discount *Monthly Discount* is a payment discount that grants an allowance of *1%* whenever the preferred way of payment *Bank Transfer* is used. For setting up these discounts, the modeling expert incorporates the intended target customer group. The expert also adds a **rating** to the promotion description that was given from an unknown former customer that reads a value of *1* and a comment *fast service* for the categories *Production & Quality* and *Responsiveness*.

Adjust Productivity and Quality Finally, the modeling expert documents productivity and quality properties that figure 8.17 presents. The MCH service defines the **quality** *New Hardware Quality*, which is strongly influ-

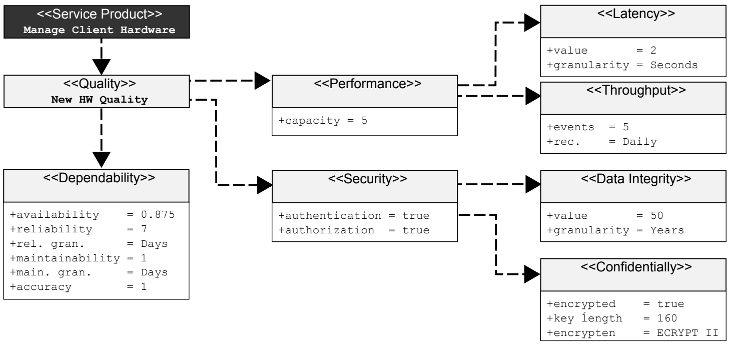


Figure 8.17.: Conceptual Service Diagram for *Manage Client Hardware* Service: Productivity & Quality.

enced by the targeted customer group and the service value level that was specified in the BSM result diagram. Moreover, the channel the service is provided over has an impact on the quality properties. The capability *Order New Hardware* refers to this quality property. Hence an assertion can be made about the capability's quality in terms of its dependability, performance, and security. The quality owns a **dependability** with an availability of *87.5%*. This is calculated with the proportion between the mean time to failure and mean time to repair (cf. subsection 5.2.8). Its reliability is set to *7 Days* as the mean time to failure and its maintainability to *1 Day* as the mean time to repair the service. The accuracy is set to *1* for no failure ever happened. Furthermore, the quality property outlines a **performance** with a capacity of *5* parallel requests, a **latency** of *2 seconds* and a **throughput** of *5 events daily*. It also depicts a **security** that provides *authentication* and *authorization* as well as a **data integrity**, which depicts *50 years* that are necessary in order to alter the data without proper authentication. Moreover, the service's data is encrypted using *ECRYPT II* and a key length of *160*.

Listing 8.5: WSDL for *Manage Client Hardware* Service: Mapping Rules 1—18

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.example.org/NewWSDLFile/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Manage_Client_Hardware" targetNamespace="http://www.example.org/NewWSDLFile/">
    <wsdl:documentation>
        [key:http://www.itcompany.com/mch]
        [description:allow outsourcing of purchasing and the maintaining of computer hardware]
        [documentation:www.documentation.de]
        [spkn l.:eng;ger;][wr l.:eng;ger;]
        [version:2]
        [created:Tue Oct 06 00:00:00 CEST 2009]
        [updated:Mon Mar 01 00:00:00 CET 2010]
        [next update:Thu Nov 10 00:00:00 CET 2011;]
        [type:Core Service]
        [automation:Partially Automated]
        [composition:Final Service]
        [customizable:false]
        [paym. cond.:http://itc.com/services/mch/paymentcondition]
        [deli. cond.:http://itc.com/services/mch/deliverycondition]
        [classi.:350097 NAICS]
        [benefit:Low Transaction Costs]
        [benefit:Low Labor Costs]
        [benefit:Low IT Costs]
        [benefit:Recent Hardware]
    </wsdl:documentation>
...
```

8.2.4. Deployment Artifacts

While the previous subsections showed the *Manage Client Hardware* service's result documents for BSM and CSM, this subsection discusses the generated WSDL file. The complete generated WSDL file is found in listing A.16. This DA's result document was automatically generated using the *Procedural QVT* transformation script introduced in section 7.3 (cf. also listing A.4).

WSDL Definition Element Listing 8.5 shows the WSDL file's initial part. Line #2 outlines the WSDL definition element with the name attribute set to *Manage_Client_Hardware*. While lines #4—#15 show attributes of the service product property that were mapped to the documentation element, lines #16—#22 list information from the terms of use, classification, and benefit properties, such as the NAICS classification in line #18.

WSDL Type Element Listing 8.6 outlines identified resources from the CSM diagram in figure 8.15. Lines #4 and #10 show the corresponding XSD elements for each resource. As aforementioned, in the following activity, IT architects need to refine these elements with concrete attributes.

Listing 8.6: WSDL for *Manage Client Hardware* Service: Mapping Rules 19—21

```
 1  ...
 2    <wsdl:types>
 3      <xsd:schema targetNamespace="http://www.example.org/
             NewWSDLFile/">
 4        <xsd:element name="Order">
 5          <xsd:annotation/>
 6          <xsd:complexType>
 7            <xsd:sequence/>
 8          </xsd:complexType>
 9        </xsd:element>
10        <xsd:element name="Hardware">
11          <xsd:annotation/>
12          <xsd:complexType>
13            <xsd:sequence/>
14          </xsd:complexType>
15        </xsd:element>
16      </xsd:schema>
17    </wsdl:types>
18  ...
```

WSDL Messages Element Listing 8.7 presents the messages part of the transformation result. According to mapping rules 22—27 one input and one output message for each capability must be generated, provided the capability features at least one pre or post condition. Figure 8.13 outlines the two capabilities *Order New Hardware* and *Return Hardware* both, with one pre condition and one post condition. Line #2 shows the input message for the former capability. The message's name equals the capability's name plus the postfix *Input*. The capability's pre condition was transformed into the message part element *New_Order* as line #3 shows. While the name attributes equal the pre condition's name, the element references refer to linked resource. The output message is done likewise to the input message, with the difference that the post conditions were considered.

Listing 8.7: WSDL for *Manage Client Hardware* Service: Mapping Rules 22—27

```
 1  ...
 2    <wsdl:message name="Order_New_HardwareInput">
 3      <wsdl:part element="tns:Order" name="New_Order"/>
 4    </wsdl:message>
 5    <wsdl:message name="Return_HardwareInput">
 6      <wsdl:part element="tns:Order" name="New_Order"/>
 7    </wsdl:message>
 8    <wsdl:message name="Return_HardwareOutput">
 9      <wsdl:part element="tns:Hardware" name="Returned_Hardware"/
          >
10    </wsdl:message>
11    <wsdl:message name="Order_New_HardwareOutput">
12      <wsdl:part element="tns:Hardware" name="New_Hardware"/>
13    </wsdl:message>
14  ...
```

WSDL portType Element Listing 8.8 presents the generated portType element that embodies information of CSM's capabilities. The portType element is named after the service product as line #2 tells. Lines #3 and #8 present WSDL operations for the service's capabilities. Both operations refer to the input and output messages that were presented in the previous paragraph.

Listing 8.8: WSDL for *Manage Client Hardware* Service: Mapping Rules 28—32

```
...
  <wsdl:portType name="Manage_Client_Hardware">
    <wsdl:operation name="Return_Hardware">
      <wsdl:documentation>Allows to return once ordered
           hardware.</wsdl:documentation>
      <wsdl:input message="tns:Return_HardwareInput"/>
      <wsdl:output message="tns:Return_HardwareOutput"/>
    </wsdl:operation>
    <wsdl:operation name="Order_New_Hardware">
      <wsdl:documentation>Form to order new hardware.</
           wsdl:documentation>
      <wsdl:input message="tns:Order_New_HardwareInput"/>
      <wsdl:output message="tns:Order_New_HardwareOutput"/>
    </wsdl:operation>
  </wsdl:portType>
...
```

8.3. Findings

After the presentation of two case studies in two different domains, this section discusses the findings.

The case studies' intention was to figure out whether the service description method supports the *documentation*, the *communication*, and the *reasoning* of service descriptions on different levels of abstractions.

The case study shows that the proposed approach is appropriate for *documenting* business service models as well as conceptual service models. In particular, the developed UML Profiles BSMN and CSMN (cf. sections 4.3 and 5.3) guarantee a full documentation of services' core ideas and their technology-agnostic conceptualization. The subject-matter-experts *business strategists* and *modeling experts*, however, were not familiar with UML or UML Profiles. This experience indicates the necessity of involving *mod-*

eling experts who are familiar with UML in order to document business and conceptual service models. One best-practice that was surfacing during the case studies is to hide the notations from subject-matter-experts during the *documentation* and rather use semi-structured interviews to elicit necessary information and use the answers to these questions in order to build business and conceptual service diagrams.

Furthermore, the case study proves that business strategists as well as marketing experts were able to use both, business and conceptual service diagrams in order to *communicate* services' main ideas with other involved subject-matter-experts. Moreover, these stakeholders were in the position to *discuss* and rethink services as well as to make informed decisions and, hence, to improve result documents.

The automatic transformation of CSM diagrams into corresponding WSDL files proved also very helpful for IT architects as a starting point for *enhancing* these artifacts for deployment.

One limitation is the resource definition and the transformation into WSDL types. CSM defines resources merely with a name and a type, but neglects to add further attributes for further resource definition, which is found in UML Class diagrams, for example. A reason for this is that data modeling is out of SDM4SE's scope. The impression during these case studies, however, is that it is eligible to link resources to *conceptual data*. A possible approach to do this is to develop a transformation script that transforms available resources from the CSM result diagram into a UML Class diagram, and then to add desirable attributes in the class diagram. This class diagram can then be used as a second input model for the transformation between CSMM and WSDL in order to generate WSDL's type element.

Part IV.

Finale

9. Related Work

After the introduction of SDM4SE in part II and its evaluation as well as the implementation in part III, this chapter presents and discusses work that is related to SDM4SE. Whereas section 2 discusses work that is important for understanding the motivation and the solution for service description, this section outlines rather approaches that address the same problem. The analysis of available work in the area of service description modeling follows eleven aspects that this thesis covers as outlined in table 9.1. The aspects *BSM*, *CSM*, and *DA* tell which levels of the refined Open-EDI reference model are covered. Next to these aspects, the aspects *method*, *notation*, and *transformation* test whether available work supports the service description modeling process. The *meta model*, *Ontology*, and *descriptive* aspects show which way of formalization was used. The final aspects tell if available work covers *functional* and *non-functional* service properties. Table 9.1 gives an overview of related work in the area of service description modeling along with the result of this analysis. The following paragraphs elaborate on each contribution.

Mylopoulos et al. 1992 [77] introduce a framework for representing as well as using non-functional requirements during the design of systems. The framework consists of five concepts: (1) goals represent non-functional requirements and (2) link types relate goals to each other, (3) methods for refining goals into fine-granulated goals, (4) correlation rules that depict goal correlation, and (5) labeling procedures that link design decisions with non-functional requirement goals. Similar to SDM4SE, Mylopoulos et al. provide a method for documenting non-functional requirements

Table 9.1.: Related Work Overview.

Author	**BSM**	**CSM**	**DA**	**Method**	**Notation**	**Transformation**	**Meta Model**	**Ontology**	**Descriptive**	**Functional**	**Non-Functional**
Mylopoulos et al. 1992 [77]		x		x					x		
Dumas et al. 2001 [35]		x							x	x	x
Oaks et al. 2003 [79]		x						x		x	
Baida et al. 2003 [15]		x						x		x	
Cysneiros and Leite 2004 [29]		x		x	x				x		x
Dobson et al. 2005 [32]			x	x				x			x
Papaioannou et al. 2006 [104]			x					x			x
O'Sullivan 2006 [99]		x						x			x
de Miranda et al. 2006 [30]		x			x			x			x
Ankolekar et al. 2008 [11]			x		x				x	x	x
Terlouw 2008 [133]			x					x		x	
Tondello & Siqueira 2008 [134]		x	x		x	x		x			x
Weigand et al. 2009 [140]	x	x		x				x		x	
Kona et al. 2009 [63]			x		x			x			

with their framework. Even though they address a conceptual level, they offer neither user support in terms of a notation nor transformations for translating conceptual requirements into software artifacts.

Dumas et al. 2001 [35] identify the need for a semantic service description framework because of the Internet's global and inexpensive connectivity. Such a conceptual description aims at non-functional properties including advertising, locating, analyzing, and comparison of services. The authors' intention is to define requirements for future service descriptions. They propose the following service characteristics: provider, availability (time &

spatial), channel, pricing, payment, security, quality of service, and reputation. The authors find the UDDI TModel appropriate as an underlying model for these service characters. Also, Dumas et al. develop an orthogonal model for service classification along the dimensions of service automation and service composition. Next to this model, the authors also define recipients of service, relationships between the service providers and their users, and the nature of demand and supply for a service and service delivery. Likewise to the SDM4SE, Dumas et al. aim at service proposition. However, they mainly propose requirements for service propositions that relate to the Conceptual Service Model. They neither give information about a modeling notation nor a procedure model.

Oaks et al. 2003 [79] discuss about the lack to specify service capabilities, that is, what services, or agents, can do. They offer a structured and machine interpretable capability description. In doing so, the authors offer an ontology for describing services' functionality in a formal manner. In detail, the ontology describes services' actions, inputs and outputs, pre- and postconditions, and domains. Each capability declares a set of inputs and outputs. Input refers to the type of information, the state of the information system (pre-condition), and the state of the domain and context of the information system (assumption), which is required in order to carry out the capability. Output refers to the type of information, the state of the information system (post-condition), and the state of the domain and context of the information system (effect), which is guaranteed after the execution of the capability. Similar to SDM4SE, Oaks et al. address a conceptual level and offer a formalization for service properties. Their approach, however, does not include a link to business models and no grounding with technical artifacts. Moreover, they offer no modeling support in terms of transformations, method, or a notation. Furthermore, the proposed ontology is limited to functional properties.

Baida et al. 2003 [15] argue that electronic commerce is still mainly characterized by the relatively straightforward trading of commodity goods. Current challenges are advanced business scenarios, such as collaborative design over the Internet of sophisticated goods and services. Their work elaborates on further challenges in order to achieve collaborative electronic commerce concerned with real-world services. Similar to the SDM4SE, Baida et al. focus on service trade and propose a knowledge structure in form of a service ontology. The differences lie in that Baida et al. restrict their service ontology to the Conceptual Service Model and functional service properties, and neither propose a modeling notation nor a guiding method.

Cysneiros and Leite 2004 [29] argue that non-functional requirements have a menial role in software design. In order to tackle this observation, they introduce a process for eliciting non-functional requirements for software design and link them to functional requirements. In order to do so, they treat non-functional requirements as goals which can be decomposed into fine-grained goals. Furthermore, for documenting elicited requirements, the authors define extensions for UML. Similar to SDM4SE, Cysneiros and Leite offer a method for documenting conceptual properties with the use of UML. While they do not define service properties themselves, they rely on the work of Mylopoulos et al. 1992 [77]. Unlike SDM4SE, the authors focus on software design and neglect business models as a valid source for non-functional requirements.

Dobson et al. 2005 [32] define a Quality of Service (QoS) ontology for service-centric systems. For them, QoS includes all non-functional properties of a service. They argue that with distributed systems and their services as black boxes, QoS properties are the key for service identification. Hence, a common way for defining QoS is necessary. Dobson et al. ground their ontology on OWL. In particular, they intend to extend OWL-

S Profile for defining non-functional properties. For doing that, they offer three different layers for applying ontologies. The base layer comprises the base QoS ontology and unit ontologies. The attribute layer contains concrete QoS attributes which are defined over the base QoS and Units ontologies. Finally, the domain-specific layer links attributes to specific domains. The QoS base ontology and its OWL-S integration is discussed in more detail. The ontology comprises the classes: (1) QoSAttribute, (2) (Un)Measurable Attribute, (3) Metric, (4) PhysicalQuantity, (5) Units, and (6) ConversionRate. The authors show how to build attributes with this ontology and how to map it to the OWL-S Profile. They introduce a tool for service discovery and QoS specification. Similar to SDM4SE, Dobson et al. acknowledge the necessity of a common way for describing services. Furthermore, the authors offer a method for defining non-functional properties. However, they rather address the Deployment Artifact level and restrict their solution to non-functional properties.

Papaioannou et al. 2006 [104] present a generic ontology for representing non-functional properties. The approach lists two parts: firstly an ontology for representing non-functional properties and secondly a language for defining them. The ontology lists the following basic concepts: QoSParameter, Metric, QoSImpact, Type, Nature, Aggregated, Node, and Relationship. The QoS Language defines these properties: Accessibility, Availability, Capacity, Scalability, Performance (Jitter, ErrorRate, Latency, Throughput), ResponseTime, Cost, Configuration (supportedStandards), and Reliability. Papaioannou et al. classify their ontology on the Deployment Artifacts level and neglect links to a conceptual level. Though they offer an ontology as a formalization, the authors provide no method nor means for functional service properties.

O'Sullivan 2006 [99] refers to non-functional properties as constraints over the functionality of services. They argue that non-functional properties of services are an essential ingredient of service descriptions. Non-functional properties (1) improve service discovery (higher degree of expressiveness), (2) allow service substitution (better service comparison), (3) advance service composition, and (4) permit service management in terms of monitoring and controlling services. Similar to the functional behavior, non-functional properties require a formal representation and a shared understanding between different parties. O'Sullivan et al. [100] offer a formal description for non-functional properties. They use Object Role Modeling (ORM) [47] for formalizing 14 different categories of non-functional properties. Their approach addresses services, which are consumed electronically (e-Services) as well as services which are consumed in a traditional manner. Similar to SDM4SE, O'Sullivan acknowledges a common set of properties for describing services. The author offers a formalization using an ontology, but refrains from defining a notation for it. Also, O'Sullivan addresses the conceptual level, however, neglects an integration for business models and deployable artifacts. Furthermore, the presented approach offers no method for modeling support.

de Miranda et al. 2006 [30] target service bundles, where service bundles refer to a set of complementary services that together fulfill a complex need. De Miranda et al. focus on pricing models on a conceptual level and formalize these by extending a previous developed ontology. While they utilize the e^3-Value approach as a notation for their work, they offer no guidance for developing such ontologies but present valid examples for their work. Also, no means for reusing pricing information in implementation artifacts exists.

Ankolekar et al. 2008 [11] present a policy-based approach for specifying preferences on web service properties. They say that service-orientation and web services need a technical infrastructure to find, bind, and execute distributed services. For service identification, the authors find functional and non-functional properties suitable. For this, Ankolekar et al. define six requirements spanning functional and non-functional properties, constraint combination, and soft-constraints. They argue that existing policy languages to define such properties are not capable to fulfill the aforementioned requirements. Following this, they define a policy that addresses these requirements. Ankolekar et al. address mainly the technical level. However, they offer no real formalization for their defined properties, nor any guidance for modeling.

Terlouw 2008 [133] finds the UDDI specification too technology-driven for specifying services and hence believes that it contradicts SOA promises of increased flexibility of service reuse and business-IT alignment. Terlouw claims that for business process execution suitable services need to be identified as well as to specified. Service registries store these specifications for identification. In consequence, she proposes the Enterprise Ontology and the business component specification for business task specification. Terlouw addresses mainly Open EDI's technical level. She formalizes her approach using an extension for an existing ontology. Furthermore, she focuses on functional properties but offers no modeling support.

Tondello and Siqueira 2008 [134] present an ontology for QoS modeling. The authors claim that this ontology may be used during service design (UML Profile) and runtime (OWL-S). Consequently, they follow OMG's [92] QoS Meta model with the packages: characteristics, dimensions, and contexts. Also, they define an extension for their ontology to combine it with OWL-S. Similar to SDM4SE, Tondello and Siqueira establish a link between a conceptual level and deployable IT artifacts, but without establishing a link to business model concepts. As a formalization, the authors chose an ontology language. Furthermore, they offer modeling support in terms of a modeling notation and a transformation concept. Unlike SDM4SE, their approach is limited to functional properties and it neglects a method for modeling guidance.

Weigand et al. 2009 [140] introduce a unified view on services by means of a service model and a modeling method for reasons of service design and analysis. They argue that there exists a business view on services, such as in the approaches from Gordijn and Osterwalder; and that a software view on services exists, namely the SOMA approach from IBM. The authors find that a gap exists between these two views and that a service model closes this gap. Following that, Weigand et al. discuss business modeling with the REA and e^3 Value Ontology, and Spohrer's service systems theory [129]. They then introduce the running example taken from the IBM SOMA paper. The service model comprises of a service ontology, a service classification, and a service layer architecture. Firstly, Weigand et al. specify five service characteristics that they found throughout service literature:

- a service is an economic resource
- a service is provided by one actor for the benefit of another
- a service existence depends on the process in which it is produced and consumed (a service is consumed and produced at the same time)
- a service encapsulates a set of resources owned by a service provider
- a service is always governed by a policy

The service ontology builds on top of the REA ontology and extends REA with the concepts: service type, work process type, and policy type. The service classification differentiates the following services: complementary services, enhancing services, supporting services, and coordination services. The service layer architecture taken from the enterprise ontology comprises three different layers for services: business services at the business social level, informational web services at the information level, and utility services at the infrastructure level. The service identification method comes with three steps: (1) value model creation or adaption (e^3 Value Ontology), (2) business service identification (service model), and (3) software service identification. Likewise to SDM4SE, Weigand et al. propose a knowledge structure in form of an ontology and utilize the e^3 Value ontology as a notation that spans business models as well as conceptual models. Their approach differs from SDM4SE in that it aims at service identification and offers no transformation of modeling artifacts. Additionally, Weigand et al. focus merely on functional properties.

Kona et al. 2009 [63] argue that automatical web service discovery, deployment, composition enables the full potential of web service technology. They say WSDL merely covers services' syntactic descriptions without any semantic grounding. The authors tackle this issue by presenting a novel language for describing services in a formal fashion. They build the Universal Service-Semantic Description Language (USDL) on top of the Web Ontology Language (OWL) and incorporate WordNet for dissolving meanings of service operations. The intended field of application is to extend existing WSDL documents with USDL. USDL spans concepts, affects, conditions and constraints, messages, and port types. In comparison to SDM4SE, Kona et al. establish a formal language which complements WSDL as a deployment artifact. They formalize their language with a well-accepted ontology language, and thus, users may utilize existing ontology tools for adopting USDL. The language, however, does not specify specific service properties. Rather, USDL offers a way to implement service properties for web services. Also, Kona et al. provide neither a development method nor a transformation support that aligns WSDL and USDL files.

10. Conclusion and Future Work

Following the discussion of work that is related to SDM4SE, this chapter concludes this thesis and offers prospects for future work. As stated in chapter 1, Globalization, technological change as well as an increasing demand for services drives countries from industrial economies toward service economies [106]. This development has an impact, next to other aspects, on how companies create value and trade services. Companies concentrate on core competencies [69, 126] for efficiency [8, 110], cost savings [8, 68], and new business opportunities [141]. This leads to the development of service ecosystems as an evolution of service orientation that makes services available as tradable products on service market places. Success factors for the development of such market places are the need for standardization [9, 126] and a strong alignment between business and IT [68, 110, 136].

One impediment for service ecosystems is a missing way of describing services in common. Such a service description would enable service proposition as well as discovery and selection. From the perspective of service providers, a business-orientation of service descriptions becomes a crucial part in the service development process, which is impeded for the following reasons. Firstly, there does not exists a formalism for defining service descriptions on a conceptual level [35, 67, 126]. Secondly, service descriptions embody divergent information and need the involvement of different subject-matter-experts. Thirdly, ample technical specifications exist that describe web services with overlapping domains, which employ first-order logic, predicates, and XML, such as Web Service Description Language (WSDL) [27], Web Service Modeling Ontology (WSMO) [112], and Semantic Annotations for WSDL and XML Schema (SA-WSDL) [37]. Fourthly, there is no real alignment between service business models and IT-related service descriptions. These reasons indicate that the service description development process is prone to errors, slow, and irreproducible.

This work looked into service descriptions and the aforementioned impediments for a business-oriented development of service descriptions. As outlined in section 1.2, the question that this thesis attempted to answer is how to describe services in a business-oriented fashion in order to leverage service proposition and service discovery. The overall result for describing services in a business-oriented fashion lies in the development of the Service Description Method for Service Ecosystems (SDM4SE) that chapter 3 presented. The method consists of a layer concept, a method engineering approach, and transformations.

The layer concept builds upon the Open-EDI reference model [55] and distinguishes between four layers. Service properties in the *Business Service Model* layer own a strategic semantics and take into account services' final purpose and context. The next layer, the *Conceptual Service Model*, represents the actual modeling purpose of service descriptions. Service properties on this layer reflect a firm establishment with concrete values. The result is a value proposition toward potential customers. *Deployment Artifacts* describe technical specifications to implement service properties. Each layer features the artifacts from method engineering: activities, roles, techniques, result documents, tools, and meta models. The last layer, service environment, concentrates on runtime as well as execution, and hence, is out of scope of a service description development process.

The method's main objective is to derive technical service descriptions from business models. And in doing so, the service description layers offer an appropriate work-break-down structure in order to reduce complexity and to establish a bridge between business and IT. The definition of method engineering artifacts provides a conceptual formalism for service descriptions. Figure 3.1 showed that method engineering artifacts need to be defined for each layer. This is due to the fact that each layer presents a discreet phase in the service description development process. By defining the method engineering artifacts for each layer, it is possible to acknowledge different subject-matter-experts involved in describing services by codifying best-practices, to manage and generate IT specifications, and to offer cohesion between business and IT, which in turn results in less errors, fasten the development process, and makes it comprehensible.

The main question, however, was subdivided into three questions, which are outlined and discussed in the following paragraphs.

[RQ1] Which service properties are relevant for service ecosystems? In order to select valid service properties, an investigation and analysis of available literature proposed properties in the disciplines of business science, information systems, and computer science. The result depicts 39 service properties and their relationships. For a better understanding and readability these properties are grouped into the following categories: (1) product, (2) process, (3) people, (4) physical evidence, (5) place and time, (6) price, (7) promotion, and (8) productivity and quality (cf. chapter 5). Section 5.2 showed a formalization of these 39 service properties using the Meta Object Facility of OMG [84] in order to foster a formal understanding of service descriptions. Listing A.2, on the other hand, showed the corresponding XMI code for this meta model.

This formal set of service properties implies a common understanding of services and offers service providers and consumers a unified language to propose and to discover services, respectively.

[RQ2] How are service properties to be modeled? Business modeling is a discipline that uses graphical languages, e.g. UML and BPMN, to elicit, to document, to communicate, and to reason about business requirements [53]. That is particularly important for business modeling usually involves many experts with different backgrounds and expertise. A dedicated modeling notation for service properties enables the application of a common set of properties. Based on the conceptual service description meta model, section 5.3 defines a UML Profile. UML Profile is part of the UML specification and offers a standard way to customize UML diagrams to cover domain-specific semantics. This enables practitioners, who are already familiar with UML, to model specific domains. Sections 8.1.3 and 8.2.3 showed the application of the developed UML Profile for describing services in the insurance and the IT outsourcing domain, respectively.

The implication of a corresponding modeling notation for the aforementioned set of service properties enables a technology-independent documentation of service descriptions. Furthermore, it enables subject-matter experts as in business strategists as well as marketing experts to participate in the service description development process.

[RQ3] How are standard web service artifacts generated from service properties? While numerous technical languages to implement web service descriptions exist, their development is not aligned with business requirements, and is slow as well as incomprehensible [124]. Section 6.1 introduced standard web services artifacts for describing services in a technical manner. Next to WSDL [27] and UDDI [96] these included the semantic concepts WSMO [112], OWL-S [74], and SA-WSDL [37]. Section 6.2 showed an abstract mapping between the conceptual service meta model and WSDL in order to show their relationship as well as to provide a means for generating WSDL automatically. It is abstract in that it merely shows possible relationships between CSM and WSDL. The abstract mapping consists of 32 mapping rules. Section 7.3, however, presented an implementation of this abstract mapping with model transformation technology.

This abstract mapping and its implementation with model-to-model technology abstracts from the complexity of generating and maintaining IT artifacts. Additionally, this approach bridges potentially different mind sets [125] between marketing experts and IT architects.

Two case studies show the proposed service description method's applicability in real-world settings. The case studies' intention was to figure out whether the service description method supports the *documentation*, the *communication*, and the *reasoning* of service descriptions on different levels of abstractions.

The case studies show that the proposed approach is appropriate for *documenting* business service models as well as conceptual service models. In particular, the developed UML Profiles BSMN and CSMN (cf. sections 4.3 and 5.3) guarantee a full documentation of services' core ideas and their technology-agnostic conceptualization. The subject-matter-experts *business strategists* and *modeling experts*, however, were not familiar with UML or UML Profiles. This experience indicates the necessity of involving *modeling experts* who are familiar with UML in order to document business and conceptual service models. One best-practice that was surfacing during the case studies is to hide the notations from subject-matter-experts during the *documentation* and rather use semi-structured interviews to elicit necessary information and use the answers to these questions in order to build business and conceptual service diagrams.

Furthermore, the case studies prove that business strategists as well as marketing experts were able to use both business and conceptual service diagrams in order to *communicate* services' main ideas with other involved subject-matter-experts. Moreover, these stakeholders were in the position to *discuss* and rethink services as well as to make informed decisions and hence, to improve result documents.

The automatic transformation of CSM diagrams into corresponding WSDL files proved also very helpful for IT architects as a starting point for *enhancing* these artifacts for deployment.

One limitation is the resource definition and the transformation into WSDL types. CSM defines resources merely with a name and a type, but neglects to add further attributes for further resource definition, which is found in UML Class diagrams, for example. A reason for this is that data modeling is out of SDM4SE's scope. The impression during these case studies, however, is that it is eligible to link resources to *conceptual data*. A possible approach to do this is to develop a transformation script that transforms available resources from the CSM result diagram into a UML Class diagram, and then to add desirable attributes in the class diagram. This class diagram can then be used as a second input model for the transformation between CSMM and WSDL in order to generate WSDL's type element.

Although the three research questions have been answered, there are still possible prospects for future work.

Next to use SDM4SE and to refine service descriptions toward deployment artifacts, there exist two other promising applications for IT: (1) modeling service-oriented architectures and (2) service engineering. Arsanjani et al. [12] define Service Oriented Modeling & Architecture (SOMA) as *"...an end-to-end software development method for building SOA-based solutions"*. This method applies to establish a design and implementation for service-oriented architectures. It specifies a life-cycle comprising 21 steps, which are grouped into seven phases: (1) business modeling & transformation, (2) solution management, (3) identification, (4) specification, (5) realization, (6) implementation, (7) deployment, monitoring, and management. The authors recognize the first phase *business modeling and transformation* as an important first step that serves as the entry point for the phase *identification* & *specification*. However, Arsanjani et al. do not further describe this phase. As a suggestion, the business service meta model and the business service modeling notation may be applied to this phase.

Contrary to SOMA, Kett et al. [62] specify the Integrated Service Engineering (ISE) Framework for developing single business services. The ISE Framework is an orthogonal matrix and similar to the Zachman framework. The vertical axis shows four perspectives of the engineering process and is named *service perspectives*. Each perspective relates to a specific role with appropriate skills and offers different sets of tools and methods. It also implies the chronology of the framework. The horizontal axis shows five different *descriptions of a service*. Each description is valid for each perspective. Each intersection in the matrix is placeholder for a meta model, a notation, and activities, which are appropriate for the respective perspective and the modeling aspect. BSM and CSM with their meta models and notations fit the ISE framework's *service* description for the *strategic perspective* and the *conceptual perspective*, respectively.

Although chapter 8 presents two case studies in different domains, further case studies need to verify the completeness of discovered service properties as well as the applicability of the whole method that aligns business models with IT. A prospect that came up during the case studies was whether to allow domain-specific extensions for CSMM. Currently, CSMM holds domain-independent service properties. Domain-specific extensions would enable amended service proposition and discovery.

Sections 6.2 and 7.3 show the abstract and the implemented transformation between CSMM and WSDL, respectively. A broader acceptance of SDM4SE requires further transformations between the conceptual service model and technical specifications. Section 6.1 lists valid technical specifications which fit deployable artifacts. These include next to WSDL, UDDI, OWL-S, WSMO, and SA-WSDL.

As aforementioned, data as well as process modeling is out of SDM4SE's scope. Nevertheless, the next steps should include an integration of SDM4-SE with data and process modeling tools, techniques, and methods. The conceptual service meta model outlines a resource type which serves as either input or output to services' capabilities. It is, however, not intended to refine resources with attributes. An integration with data models would allow to specify a reference to a data entity in a data model for refinement. The same holds true for the capability type that refer to services' functionality for service consumers. Nevertheless, the capability type is an interface for a service process that describes capabilities' behaviors. A reference from capabilities to process models enables the specification of such behaviors during the service engineering process.

Another possibility that came up during the case studies was to process existing service repositories and to generate a CSM diagram for each WSDL file. This would allow further reasoning about existing web services and whether they embody potential for reuse as well as to offer them on service market places.

Bibliography

[1] Eclipse Model Development Tools (MDT). http://www.eclipse.org/modeling/mdt/. last accessed: April 14, 2010.

[2] Eclipse Model to Text (M2T). http://www.eclipse.org/modeling/m2t/. last accessed: April 14, 2010.

[3] Eclipse modeling project (emp). http://www.eclipse.org/modeling/. last accessed: April 14, 2010.

[4] Gmf tutorial. http://wiki.eclipse.org/GMF_Tutorial. last accessed: April 14, 2010.

[5] Theseus TEXO – Business Webs in the Internet of Services. http://www.theseus-programm.de/en-us/theseus-application-scenarios/texo/default.aspx. last accessed: 2009-03-05.

[6] The Dublin Core Metadata Initiative. http://purl.oclc.org/dc/, 1998.

[7] AL-MASRI, E. The QWS Dataset. http://www.uoguelph.ca/ qmahmoud/qws/. last accessed, 2009-12-04.

[8] ALT, R., FLEISCH, E., AND WERLE, O. The Concept of Networkability - How to Make Companies Competitive in Business Networks. In *ECIS* (2000).

[9] ALT, R., REICHMAYR, C., PUSCHMANN, T., LESER, F., AND ÖSTERLE, H. An Engineering Approach to Develop Business Networks. In *I3E* (2001), pp. 209–228.

[10] ALVES, A., ARKIN, A., ASKARY, S., BARRETO, C., BLOCH, B., CURBERA, F., FORD, M., GOLAND, Y., GUIZAR, A., KARTHA, N., AND LIU, C. K. Specification: Business Process Execution Language for Web Services version 2.0, January 2007.

[11] ANKOLEKAR, A., LAMPARTER, S., AND STUDER, R. Semantic specification and evaluation of bids in web-based markets. *Electronic Commerce Research and Applications 7*, 3 (2008), 313–329.

[12] ARSANJANI, A., GHOSH, S., ALLAM, A., ABDOLLAH, T., GARIAPATHY, S., AND HOLLEY, K. SOMA: a method for developing service-oriented solutions. *IBM Syst. J. 47*, 3 (2008), 377–396.

[13] AVIZIENIS, A., LAPRIE, J.-C., RANDELL, B., AND LANDWEHR, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing 1*, 1 (2004), 11–33.

[14] BAIDA, Z., AKKERMANS, H., AND BERNARAS, A. The Configurable Nature of Real-World Services: Analysis and Demonstration, Aug. 23 2003.

[15] BAIDA, Z., AKKERMANS, H., AND GORDIJN, J. Serviguration: towards online configurability of real-world services. In *ICEC* (2003), N. M. Sadeh, M. J. Dively, R. J. Kauffman, Y. Labrou, O. Shehory, R. Telang, and L. F. Cranor, Eds., vol. 50 of *ACM International Conference Proceeding Series*, ACM, pp. 111–118.

[16] BAIDA, Z., GORDIJN, J., AND OMELAYENKO, B. A shared Service Terminology for Online Service Provisioning. In *ICEC* (2004), M. Janssen, H. G. Sol, and R. W. Wagenaar, Eds., vol. 60 of *ACM International Conference Proceeding Series*, ACM, pp. 1–10.

[17] BARBACCI, M., KLEIN, M. H., LONGSTAFF, T. A., AND WEINSTOCK, C. B. Quality Attributes. Tech. Rep. ESC-TR-95-021, Software Engi-

neering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, December 1995.

[18] BARROS, A. P., AND DUMAS, M. The Rise of Web Service Ecosystems. *IT Professional 8*, 5 (2006), 31–37.

[19] BARROS, A. P., DUMAS, M., AND TER HOFSTEDE, A. H. M. Service interaction patterns. In *Business Process Management* (2005), W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, Eds., vol. 3649, pp. 302–318.

[20] BATTLE, S., BERNSTEIN, A., BOLEY, H., GROSOF, B., GRUNINGER, M., HULL, R., KIFER, M., MARTIN, D., MCILRAITH, S., MCGUINNESS, D., ET AL. Semantic Web Services Framework (SWSF) Overview, September 2005.

[21] BICER, V., BOGERT, S., WINKLER, M., SCHEITHAUER, G., VOIGT, K., CARDOSO, J., AND AITENBICHLER, E. Modeling Services using ISE Framework: Foundations and Extensions. In *Modern Software Engineering Concepts and Practices: Advanced Approaches*, A. H. Dogru and V. Bicer, Eds. IGI Global, 2010, ch. 6.

[22] BOOMS, B., AND BITNER, M. Marketing strategies and organization structures for service firms. *Marketing of Services, American Marketing Association, Chicago, IL* (1981), 47–51.

[23] BRISCOE, G., AND DE WILDE, P. Digital Ecosystems : Evolving Service-orientied Architectures.

[24] BUXMANN, P., HESS, T., AND LEHMANN, S. Software as a service. *Wirtschaftsinformatik 50*, 6 (2008), 500–503.

[25] CHANG, E., AND WEST, M. Digital Ecosystems A Next Generation of the Collaborative Environment. In *iiWAS* (2006), pp. 3–24.

[26] CHINNICI, R., MOREAU, J.-J., RYMAN, A., AND WEERAWARANA, S. Specification: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation, 6 2007.

[27] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March, 15 2001.

[28] CHUNG, L., NIXON, B. A., AND YU, E. Using Quality Requirements to Systematically Development Quality Software. In *Proc. 4th Int. Conf Software Quality* (3–5 Oct. 1994).

[29] CYSNEIROS, L. M., AND DO PRADO LEITE, J. C. S. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Trans. Software Eng. 30*, 5 (2004), 328–350.

[30] DE MIRANDA, B., BAIDA, Z., AND GORDIJN, J. Modelling Pricing for Configuring e-Service Bundles. In *19th Bled eConference - Research Volume* (Bled, Slovenia, June 5-7 2006).

[31] DECKER, G., AND BARROS, A. Interaction modeling using bpmn. In *Proceedings of the 1st International Workshop on Collaborative Business Processes (CBP)* (Brisbane, Australia, September 2007).

[32] DOBSON, G., LOCK, R., AND SOMMERVILLE, I. QoSOnt: a QoS Ontology for Service-Centric Systems. In *EUROMICRO-SEAA* (2005), IEEE Computer Society, pp. 80–87.

[33] DORN, J., GRUN, C., WERTHNER, H., AND ZAPLETAL, M. A Survey of B2B Methodologies and Technologies: From Business Models towards Deployment Artifacts. In *HICSS* (2007), IEEE Computer Society, p. 143.

[34] DUBRAY, J.-J., AMAND, S. S., AND MARTIN, M. J. Specification: ebXML Business Process Specification Schema Technical Specification v2.0.4. Tech. rep., OASIS, December 2006.

[35] DUMAS, M., O'SULLIVAN, J., HERAVIZADEH, M., EDMOND, D., AND TER HOFSTEDE, A. H. M. Towards A Semantic Framework for Service Description. In *DS-9* (2001), pp. 277–291.

[36] FABRO, M. D. D., BEZIVIN, J., JOUAULT, F., BRETON, E., AND GUELTAS, G. Amw: a generic model weaver. In *Proceedings of the 1ere Journee sur l'Ingenierie Dirigee par les Modeles (IDM05)* (2005).

[37] FARRELL, J., AND LAUSEN, H. Specification: Semantic Annotations for WSDL and XML Schema (SA-WSDL). http://www.w3.org/TR/sawsdl/, August, 28 2007.

[38] FENSEL, D., AND BUSSLER, C. The web service modeling framework WSMF. *Electronic Commerce Research and Applications 1*, 2 (2002), 113–137.

[39] GAO, S., SPERBERG-MCQUEEN, C. M., AND THOMPSON, H. S. Specification: W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. http://www.w3.org/TR/xmlschema11-1, April 2009.

[40] GIACHETTI, G., MARÍN, B., AND PASTOR, O. Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles. In *Proceedings Advanced Information Systems Engineering, 21st International Conference, CAiSE* (Amsterdam, The Netherlands, June 8-12 2009), vol. 5565 of *Lecture Notes in Computer Science*, Springer, pp. 110–124.

[41] GIALLONARDO, E., AND ZIMEO, E. More semantics in qos matching. In *SOCA* (2007), IEEE Computer Society, pp. 163–171.

[42] GIL, A., AND BLYTHE, J. How can a structured representation of capabilities help in planning?, May 04 2000.

[43] GORDIJN, J. E^3-value in a Nutshell. Tech. rep., HEC University Lausanne, Lausanne, Oct. 07 2002.

[44] GORDIJN, J., AKKERMANS, H., AND VAN VLIET, H. Value-based Requirements Creation for Electronic Commerce Applications. *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on* (2000), 10.

[45] GRÖNROOS, C. *Service Management and Marketing - Customer Management in Service Competition*, 3 ed. John Wiley & Sons Ltd, 2007.

[46] GUTZWILLER, T. *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen.* Physica-Verlag, Heidelberg, 1994.

[47] HALPIN, T. A., AND WAGNER, G. Modeling reactive behavior in ORM. In *ER* (2003), I.-Y. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann, Eds., vol. 2813 of *Lecture Notes in Computer Science*, Springer, pp. 567–569.

[48] HEPP, M., LEUKEL, J., AND SCHMITZ, V. A Quantitative Analysis of eClass, UNSPSC, eOTD, and RNTD: Content, Coverage, and Maintenance. In *ICEBE* (2005), pp. 572–581.

[49] HEVNER, A. R., AND MARCH, S. T. The Information Systems Research Cycle. *IEEE Computer 36*, 11 (2003), 111–113.

[50] HOEKSTRA, R., BREUKER, J., BELLO, M. D., AND BOER, A. The LKIF Core Ontology of Basic Legal Concepts. In *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)* (June 2007), P. Casanovas, M. A. Biasiotti, E. Francesconi, and M. T. Sagri, Eds.

[51] HU, H., SCHEITHAUER, G., AND WIRTZ, G. ISE – Integrated Service Engineering: Applying an Architecture for Model to Model

Transformations. In *The 2010 International Conference on Software Engineering and Knowledge Engineering (SEKE'10)* (Redwood City, California, USA, July, 1 - 3 2010).

[52] IHK HANNOVER. Gründerportal der IHK. http://www.ihk-startup.de/themen-gruender/konzept.html. last accessed: 2009-10-02.

[53] INDULSKA, M., GREEN, P., RECKER, J. C., AND ROSEMANN, M. Business process modeling : perceived benefits. In *28th International Conference on Conceptual Modeling* (Gramado, Brazil, November, 9-12 2009).

[54] INMON, W. H., ZACHMAN, J. A., AND GEIGER, J. G. *Data Stores, Data Warehousing, and the Zachman Framework: Managing Enterprise Knowledge.* McGraw-Hill, Inc. New York, NY, USA, 1997. ISBN:0070314292.

[55] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). Open-edi reference model, iso standard 14662, second edition, 2004.

[56] JANIESCH, C., RUGGABER, R., AND SURE, Y. Eine Infrastruktur für das Internet der Dienste. HMD - Praxis der Wirtschaftsinformatik (45:261), 2008, pp. 71-79, June 2008.

[57] KAPLAN, R. S., AND NORTON, D. P. The balanced scorecard - measures that drive performance. *Harvard Business Review January-February* (1992), 71–79.

[58] KAVANTZAS, N., BURDETT, D., RITZINGER, G., FLETCHER, T., LAFON, Y., AND BARRETO, C. Specification: Web Services Choreography Description Language Version 1.0. Website: http://www.w3.org/TR/ws-cdl-10/, November 2005.

[59] KELLER, A., AND LUDWIG, H. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *J. Network Syst. Manage 11*, 1 (2003).

[60] KELLER, G., NÜTTGENS, M., AND SCHEER, A. W. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[61] KETT, H., SCHEITHAUER, G., WEINER, N., AND WEISBECKER, A. Integrated Service Engineering (ISE) for Service Ecosystems: An Interdisciplinary Methodology for the Internet of Services. In *Proceedings of eChallenges e-2009 Conference* (Istanbul, Turkey, October 2009), P. Cunningham and M. Cunningham, Eds., IIMC International Information Management Corporation.

[62] KETT, H., VOIGT, K., SCHEITHAUER, G., AND CARDOSO, J. Service Engineering in Business Ecosystems. In *Proceedings of the XVIII. International RESER Conference* (Stuttgart, Germany, September, 25 - 26 2008).

[63] KONA, S., BANSAL, A., SIMON, L., MALLYA, A., GUPTA, G., AND HITE, T. D. USDL: A Service-Semantics Description Language for Automatic Service Discovery and Composition. *Int. J. Web Service Res. 6*, 1 (2009), 20–48.

[64] KOTLER, P. *Marketing Management: Analysis, Planning, Implementation, and Control*, 6th ed. Prentice Hall International, 1988.

[65] KOTLER, P. *Marketing Management (Millenium Edition)*. Prentice Hall, 2000.

[66] KOTLER, P., AND KELLER, K. L. *Framework for Marketing Management (3rd Edition)*, 3rd ed. Prentice Hall, 2007.

[67] KUROPKA, D., TROEGER, P., STAAB, S., AND WESKE, M., Eds. *Semantic Service Provisioning*. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78616-0.

[68] KUTVONEN, L., METSO, J., AND RUOKOLAINEN, T. Inter-enterprise collaboration management in dynamic business networks. In *OTM Conferences (1)* (2005), R. Meersman, Z. Tari, M.-S. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H.-A. Jacobsen, J. P. Loyall, M. Kifer, and S. Spaccapietra, Eds., vol. 3760 of *Lecture Notes in Computer Science*, Springer, pp. 593–611.

[69] LAM, A., AND RAY, P. K. Security alert management in e-business networks. In *ICEB* (2004), J. Chen, Ed., Academic Publishers/World Publishing Corporation, pp. 880–885.

[70] LEE, K., JEON, J., LEE, W., JEONG, S.-H., AND PARK, S.-W. Qos for web services: Requirements and possible approaches, November 2003.

[71] LOVELOCK, C., AND WRIGHT, L. *Principles of Service Marketing and Management*, 2 ed. Prentice Hall, 2002.

[72] LUOMA, E., AND VAHTERA, H. Current and emerging requirements for digital rights management systems through examination of business networks. In *HICSS* (2004).

[73] LUSCH, R., AND VARGO, S. *The Service-dominant Logic of Marketing: Dialog, Debate, And Directions*. M.E. Sharpe, 2006.

[74] MARTIN, D. L., PAOLUCCI, M., MCILRAITH, S. A., BURSTEIN, M. H., MCDERMOTT, D. V., MCGUINNESS, D. L., PARSIA, B., PAYNE, T. R., SABOU, M., SOLANKI, M., SRINIVASAN, N., AND SYCARA, K. P. Bringing Semantics to Web Services: The OWL-S Approach. In *SWSWPC* (2004), J. Cardoso and A. P. Sheth, Eds., vol. 3387 of *Lecture Notes in Computer Science*, Springer, pp. 26–42.

[75] MOMOTKO, M., GAJEWSKI, M., LUDWIG, A., KOWALCZYK, R., KOWALKIEWICZ, M., AND ZHANG, J. Y. Towards adaptive management of QoS-aware service compositions. *Multiagent and Grid Systems 3*, 3 (2007), 299–312.

[76] MÖRSCHEL, I. C., AND HÖCK, H. Grundstruktur für die Beschreibung von Dienstleistungen in der Ausschreibungsphase. Beuth Verlag GmbH, 2001. Ref.Nr. PAS1018:2002-12.

[77] MYLOPOULOS, J., CHUNG, L., AND NIXON, B. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering 18*, 6 (1992), 483–497.

[78] NORMANN, R. *Reframing business: When the map changes the landscape.* John Wiley & Sons Ltd, 2001.

[79] OAKS, P., TER HOFSTEDE, A. H. M., AND EDMOND, D. Capabilities: Describing What Services Can Do. In *ICSOC* (2003), M. E. Orlowska, S. Weerawarana, M. P. Papazoglou, and J. Yang, Eds., vol. 2910 of *Lecture Notes in Computer Science*, Springer, pp. 1–16.

[80] OASIS EBXML COLLABORATION PROTOCOL PROFILE AND AGREEMENT TECHNICAL COMMITTEE. Specification: Collaboration-protocol profile and agreement, version 2.0. http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf, September 2002.

[81] OBJECT MANAGEMENT GROUP (OMG). Specification: Model Driven Architecture (MDA). http://www.omg.org/mda/.

[82] OBJECT MANAGEMENT GROUP (OMG). Object Constraint Language (OCL), version 2.0. http://www.omg.org/spec/OCL/2.0/, May 2006.

[83] Object Management Group (OMG). Specification: Business Process Modeling Notation (BPMN), Version 1. http://www.omg.org/docs/dtc/06-02-01.pdf, Feb 2006.

[84] Object Management Group (OMG). Specification: Meta Object Facility (MOF), Version 2.0. http://www.omg.org/spec/MOF/2.0/PDF, January 2006.

[85] Object Management Group (OMG). Specification: UML Profile and Metamodel for Services RFP UPMS "Services Metamodel". http://www.omg.org/cgi-bin/doc?soa/2006-09-09, September 2006.

[86] Object Management Group (OMG). Specification: MOF 2.0/XMI Mapping (XMI), Version 2.1.1. http://www.omg.org/spec/XMI/2.1.1/PDF, December 2007.

[87] Object Management Group (OMG). Specification: Unified Modeling Lanugage (UML) version 2.0. http://www.omg.org/docs/formal/05-07-04.pdf, July 2007.

[88] Object Management Group (OMG). Specification: MOF Model to Text Transformation Language (MOFM2T), Version 1.0. http://www.omg.org/spec/MOFM2T/1.0/PDF, January 2008.

[89] Object Management Group (OMG). Specification: Query/View/Transformation (QVT). http://www.omg.org/spec/QVT/, April 2008.

[90] Object Management Group (OMG). Specification: Semantics of Business Vocabulary and Rules (SBVR), Version 1.0. http://www.omg.org/spec/SBVR/1.0/, January 2008.

[91] Object Management Group (OMG). Specification: Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS). http://www.omg.org/docs/ad/08-08-04.pdf, August 2008.

[92] OBJECT MANAGEMENT GROUP (OMG). Specification: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, v 1.1. http://www.omg.org/technology/documents/formal/QoS_FT.htm, April 2008.

[93] OBJECT MANAGEMENT GROUP (OMG). Specification: Business Process Modeling Notation (BPMN), Version 2.0. http://www.omg.org/docs/dtc/09-08-14.pdf, Aug 2009.

[94] OBJECT MANAGEMENT GROUP (OMG). Specification: Systems Modeling Language (SysML). http://www.omg.org/spec/SysML/1.2/, June 2010.

[95] O'BRIEN, L., MERSON, P., AND BASS, L. Quality Attributes for Service-Oriented Architectures. In *SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments* (Washington, DC, USA, 2007), IEEE Computer Society, p. 3.

[96] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS). Specification: Universal Description Discovery and Integration (UDDI) Version 3.0. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm, April 2004.

[97] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS). Specification: Reference Model for Service Oriented Architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, October 2006.

[98] OSTERWALDER, A. *The Business Model Ontology: A Proposition in a Design Science Approach.* PhD thesis, Universite de Lausanne Ecole des Hautes Etudes Commerciales, 2004.

[99] O'SULLIVAN, J. *Towards a Precise Understanding of Service Properties.* PhD thesis, Queensland University of Technology, 2006.

[100] O'SULLIVAN, J., EDMOND, D., AND TER HOFSTEDE, A. Formal description of non-functional service properties. Tech. rep., Technical report, Queensland University of Technology, Brisbane, 2005. Available from http://www. service-description. com, February 2005.

[101] O'SULLIVAN, J., EDMOND, D., AND TER HOFSTEDE, A. H. M. The Price of Services. In *ICSOC* (2005), B. Benatallah, F. Casati, and P. Traverso, Eds., vol. 3826 of *Lecture Notes in Computer Science*, Springer, pp. 564–569.

[102] OUYANG, C., VAN DER AALST, DUMAS, M., TER HOFSTEDE, AND M., A. H. From business process models to process-oriented software systems: The BPMN to BPEL way. ePrint: http://eprints.qut.edu.au/archive/00005266/, October 2006.

[103] PAOLUCCI, M., KAWAMURA, T., PAYNE, T. R., AND SYCARA, K. P. Semantic matching of web services capabilities. In *International Semantic Web Conference* (2002), I. Horrocks and J. A. Hendler, Eds., vol. 2342 of *Lecture Notes in Computer Science*, Springer, pp. 333–347.

[104] PAPAIOANNOU, I. V., TSESMETZIS, D. T., ROUSSAKI, I., AND ANAGNOSTOU, M. E. A QoS Ontology Language for Web-Services. In *AINA (1)* (2006), IEEE Computer Society, pp. 101–106.

[105] PAPAZOGLOU, M. P. Service-Oriented Computing: Concepts, Characteristics and Directions. In *WISE* (2003), IEEE Computer Society, pp. 3–12.

[106] PENEDER, M., KANIOVSKI, S., AND DACHS, B. What Follows Tertiarisation? Structural Change and the Role of Knowledge-based Services. *The Service Industries Journal 23 Issue 2*, 146 (March 2003), 47–66.

[107] PIJPERS, V., AND GORDIJN, J. Bridging Business Value Models and Process Models in Aviation Value Webs via Possession Rights. In *HICSS* (2007), IEEE Computer Society, p. 175.

[108] QUARTEL, D. A. C., STEEN, M. W. A., POKRAEV, S., AND VAN SINDEREN, M. COSMO: A conceptual Framework for Service Modelling and Refinement. *Information Systems Frontiers* 9, 2-3 (2007), 225–244.

[109] RECKER, J., AND MENDLING, J. On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages, January 2006.

[110] REITBAUER, S., KOHLMANN, F., ECKERT, C., MANSFELDT, K., AND ALT, R. Redesigning business networks: reference process, network and service map. In Wainwright and Haddad [139], pp. 540–547.

[111] ROMAN, D., DE BRUIJN, J., MOCAN, A., LAUSEN, H., DOMINGUE, J., BUSSLER, C., AND FENSEL, D. WWW: WSMO, WSML, and WSMX in a Nutshell. In *ASWC* (2006), pp. 516–522.

[112] ROMAN, D., KELLER, U., LAUSEN, H., DE BRUIJN, J., LARA, R., STOLLBERG, M., POLLERES, A., FEIER, C., BUSSLER, C., AND FENSEL, D. Web Service Modeling Ontology. *Applied Ontology 1*, 1 (2005), 77–106.

[113] ROMAN, D., SCICLUNA, J., AND FEIER, C. Ontology-based choreography and orchestration of wsmo services. Website: http://www.wsmo.org/TR/d14/v0.1/, March 2005.

[114] SCHEITHAUER, G. Process-oriented Requirement Modeling for the Internet of Services. In *Proceedings of the 1st Internet of Services Doctoral Symposium 2008 (I-ESA)* (Berlin, Germany, March, 25 2008), R. Ruggaber, Ed., vol. Vol-374.

[115] SCHEITHAUER, G. Business Service Description Methodology for Service Ecosystems. In *Proceedings of the CAiSE-DC'09 16th Doctoral Consortium held in conjunction with CAiSE'09 Conference Amsterdam, The Netherlands, June 9-10, 2009* (Amsterdam, The Netherlands, June 2009), H. Weigand and S. Brinkkemper, Eds., vol. 479 of *ceur-ws*.

[116] SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. Describing Services for Service Ecosystems. In *ICSOC Workshops* (Sydney, Australia, December, 1 2008), G. Feuerlicht and W. Lamersdorf, Eds., vol. 5472 of *Lecture Notes in Computer Science*, Springer, pp. 242–255.

[117] SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. Business Modeling for Service Engineering: Toward an Integrated Procedure Model. In *Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE'2009), Boston, Massachusetts, USA, July 1-3, 2009* (Boston, MA, USA, July, 1 - 3 2009), pp. 322–327.

[118] SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. Service Value Properties for Service Ecosystems: A Reference Model and a Modeling Guideline. In *Proceedings of the EOMAS Workshop* (Amsterdam, Netherlands, June, 8-9 2009), J. Barjis, J. Kinghorn, S. Ramaswamy, E. Dubois, and P. Johannesson, Eds., vol. Vol-458 of *CEUR-WS*.

[119] SCHEITHAUER, G., KETT, H., KAISER, J., HACKNER, S., HU, H., AND WIRTZ, G. Business Modeling for Service Engineering: A Case Study in the IT Outsourcing Domain. In *SAC 2010, 25th Symposium On Applied Computing, Enterprise Engineering Track* (Sierre, Switzerland, March, 22-26 2010), pp. 118–123.

[120] SCHEITHAUER, G., VOIGT, K., BICER, V., HEINRICH, M., STRUNK, A., AND WINKLER, M. Integrated Service Engineering Workbench:

Service Engineering for Digital Ecosystems . In *Proceedings of the International ACM Conference on Management of Emergent Digital Ecosystems (MEDES)* (Lyon, France, October 2009), pp. 446–449.

[121] SCHEITHAUER, G., VOIGT, K., BICER, V., HEINRICH, M., STRUNK, A., AND WINKLER, M. ISE Workbench: Integrated Service Engineering. Business Process Management Conference Demo Track, September 2009.

[122] SCHEITHAUER, G., AND WINKLER, M. A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik 78, Bamberg University, October 2008. ISSN 0937-3349.

[123] SCHEITHAUER, G., AND WIRTZ, G. Applying Business Process Management Systems – a Case Study. In *The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE'08)* (Redwood City, California, USA, July, 1 - 3 2008), pp. 12–15.

[124] SCHEITHAUER, G., AND WIRTZ, G. Business Modeling for Service Descriptions: A Meta Model and a UML Profile. In *APCCM 2010, 7th Asia-Pacific Conference on Conceptual Modelling* (Brisbane, Australia, January, 18-21 2010), pp. 79–88.

[125] SCHEITHAUER, G., WIRTZ, G., AND TOKLU, C. Bridging the Semantic Gap between Process Documentation and Process Execution. In *The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE'08)* (Redwood City, California, USA, July, 1 - 3 2008).

[126] SCHUMACHER, N. A theoretical concept for xml-enabled global small business networks. In *ISICT* (2004), pp. 178–184.

[127] SOFTWARE ENGINEERING STANDARDS COMMITTEE OF THE IEEE COMPUTER SOCIETY USA. IEEE Guide for Software Requirements Specifications 830-1998, 1998.

[128] SOWA, J. F., AND ZACHMAN, J. A. Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal 31*, 3 (1992), 590–616.

[129] SPOHRER, J., MAGLIO, P. P., BAILEY, J., AND GRUHL, D. Steps Toward a Science of Service Systems. *IEEE Computer 40*, 1 (Jan. 2007), 71–77.

[130] STUDER, R., GRIMM, S., AND ABECKER, A. *Semantic Web Services: Concepts, Technologies, and Applications.* Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2007.

[131] SYCARA, K. P., WIDOFF, S., KLUSCH, M., AND LU, J. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems 5*, 2 (2002), 173–203.

[132] TAPSCOTT, D., TICOLL, D., AND LOWY, A. *Digital Capital: Harnessing the Power of Business Webs.* Harvard Business School Press, May 2000.

[133] TERLOUW, L. Towards a Business-Oriented Specification for Services. In *Advances in Enterprise Engineering I, 4th International Workshop CIAO! and 4th International Workshop EOMAS* (Montpellier, France, June 16-17 2008), vol. 10 of *Lecture Notes in Business Information Processing*, Springer, pp. 122–136.

[134] TONDELLO, G. F., AND SIQUEIRA, F. The QoS-MO ontology for semantic QoS modeling. In Wainwright and Haddad [139], pp. 2336–2340.

[135] VAN DER AALST, W., LEYMANN, F., AND REISIG, W. The Role of Business Processes in Service Oriented Architectures (Editorial). *International Journal of Business Process Integration and Management 2*, 2 (2007), 75–80.

[136] VAN HECK, E., AND VERVEST, P. Smart business networks: how the network wins. *Commun. ACM 50*, 6 (2007), 28–37.

[137] VARGO, S. L., AND LUSCH, R. F. Evolving to a New Dominant Logic for Marketing. *Journal of Marketing 68*, 1 (January 2004), 1–17.

[138] W3C WORKING GROUP. Web services glossary. http://www.w3.org/TR/ws-gloss/, February 2004.

[139] WAINWRIGHT, R. L., AND HADDAD, H., Eds. *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008* (2008), ACM.

[140] WEIGAND, H., JOHANNESSON, P., ANDERSSON, B., AND BERGHOLTZ, M. Value-Based Service Modeling and Design: Toward a Unified View of Services. In *CAiSE* (2009), P. van Eck, J. Gordijn, and R. Wieringa, Eds., vol. 5565 of *Lecture Notes in Computer Science*, Springer, pp. 410–424.

[141] WEISS, P., AND MAEDCHE, A. Towards adaptive ontology-based virtual business networks. In *PRO-VE* (2003), L. M. Camarinha-Matos and H. Afsarmanesh, Eds., vol. 262 of *IFIP Conference Proceedings*, Kluwer, pp. 297–304.

[142] WINKLER, M., CARDOSO, J., AND SCHEITHAUER, G. Challenges of Business Service Monitoring in the Internet of Services. In *iiWAS'2008 - The Tenth International Conference on Information Integration and Web-based Applications Services* (Linz, Austria, November, 24 - 26 2008), books@ocg.at, Austrian Computer Society, pp. 613–616.

[143] WINTER, R., KRCMAR, H., SINZ, E. J., ZELEWSKI, S., AND HEVNER, A. R. What in Fact is Fundamental Research in Business and Information Systems Engineering? *Business and Information Systems Engineering 2* (2009), 223–231.

[144] ZACHMAN, J. A. A Framework for Information Systems Architecture. *IBM Systems Journal 26*, 3 (1987), 276–292.

[145] ZACHMAN, J. A. John Zachman's Concise Definition of the Enterprise Framework. http://zachmaninternational.com/index.php/ea-articles/26-articles/27-john-zachmans-concise-definition-of-the-enterprise-framework, 2008.

[146] ZEITHAML, V. A., AND BITNER, M. J. *Service Marketing: Integrating Customer Focus Across the Firm*, 2nd ed. David Kendric Brake, 2000.

[147] ZENG, L., BENATALLAH, B., NGU, A., DUMAS, M., KALAGNANAM, J., AND CHANG, H. QoS-aware middleware for web services composition.

Part V.

Appendix

A. Listings

A.1. Ecore Representation of Meta Models

Listing A.1: BSMM Ecore

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="BusinessServiceModel"
    nsURI="http://www.siemens.com/CT/IC1/TEXO" nsPrefix="bm">
  <eClassifiers xsi:type="ecore:EEnum" name="CustomerBuyingCycle">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="CustomerBuyingCycle"/>
    </eAnnotations>
    <eLiterals name="NotDefined" literal="Not␣Defined"/>
    <eLiterals name="Awareness" value="1"/>
    <eLiterals name="Evaluation" value="2"/>
    <eLiterals name="Purchase" value="3"/>
    <eLiterals name="AfterSales" value="4" literal="After␣Sales"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="CustomerBuyingCycleObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="CustomerBuyingCycle:Object"/>
      <details key="baseType" value="CustomerBuyingCycle"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="CustomerEquity">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="CustomerEquity"/>
    </eAnnotations>
    <eLiterals name="NotDefined" literal="Not␣Defined"/>
    <eLiterals name="Acquisition" value="1"/>
    <eLiterals name="Retetion" value="2"/>
    <eLiterals name="AddonSelling" value="3" literal="Add-on␣Selling"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="CustomerEquityObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="CustomerEquity:Object"/>
      <details key="baseType" value="CustomerEquity"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="DistributionChannel">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="DistributionChannel"/>
      <details key="kind" value="empty"/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="distributionChannelID"
      lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//ID"
      iD="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="DistributionChannelID"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Name"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
```

```
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="Description"/>
          </eAnnotations>
        </eStructuralFeatures>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
              reasoning" eType="#//Reasoning">
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="Reasoning"/>
          </eAnnotations>
        </eStructuralFeatures>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
              customerBuyingCycle" eType="#//CustomerBuyingCycle">
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="CustomerBuyingCycle"/>
          </eAnnotations>
        </eStructuralFeatures>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
              valueLevel" eType="#//ValueLevel">
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="ValueLevel"/>
          </eAnnotations>
        </eStructuralFeatures>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
              priceLevel" eType="#//PriceLevel">
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="PriceLevel"/>
          </eAnnotations>
        </eStructuralFeatures>
        <eStructuralFeatures xsi:type="ecore:EReference" name="
              DC_TC" lowerBound="1" upperBound="-1"
            eType="#//TargetCustomer" resolveProxies="false">
          <eAnnotations source="http:///org/eclipse/emf/ecore/util/
                ExtendedMetaData">
            <details key="kind" value="attribute"/>
            <details key="name" value="DC_TC"/>
          </eAnnotations>
```

```
      </eStructuralFeatures>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="LifeCycleStep">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="LifeCycleStep"/>
    </eAnnotations>
    <eLiterals name="NotDefined" literal="Not␣Defined"/>
    <eLiterals name="ValueCreation" value="1" literal="Value␣Creation"/>
    <eLiterals name="ValuePurchase" value="2" literal="Value␣Purchase"/>
    <eLiterals name="ValueUse" value="3" literal="Value␣Use"/>
    <eLiterals name="ValueRenewal" value="4" literal="Value␣Renewal"/>
    <eLiterals name="ValueTransfer" value="5" literal="Value␣Transfer"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="LifeCycleStepObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="LifeCycleStep:Object"/>
      <details key="baseType" value="LifeCycleStep"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="PriceLevel">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="PriceLevel"/>
    </eAnnotations>
    <eLiterals name="NotDefined" literal="Not␣Defined"/>
    <eLiterals name="Free" value="1"/>
    <eLiterals name="Economic" value="2"/>
    <eLiterals name="Market" value="3"/>
    <eLiterals name="HighEnd" value="4" literal="High-End"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="PriceLevelObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="PriceLevel:Object"/>
      <details key="baseType" value="PriceLevel"/>
    </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="PricingMethod">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
        <details key="name" value="PricingMethod"/>
      </eAnnotations>
      <eLiterals name="NotDefined" literal="Not␣Defined"/>
      <eLiterals name="FixedPrice" value="1" literal="Fixed␣Price
            "/>
      <eLiterals name="DifferentialPrice" value="2" literal="
            Differential␣Price"/>
      <eLiterals name="MarketbasedPrice" value="3" literal="
            Market-based␣Price"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EDataType" name="
          PricingMethodObject" instanceClassName="org.eclipse.emf.
          common.util.Enumerator">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
        <details key="name" value="PricingMethod:Object"/>
        <details key="baseType" value="PricingMethod"/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="Reasoning">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
        <details key="name" value="Reasoning"/>
      </eAnnotations>
      <eLiterals name="NotDefined" literal="Not␣Defined"/>
      <eLiterals name="ServiceUsage" value="1" literal="Service␣
            Usage"/>
      <eLiterals name="RiskReduction" value="2" literal="Risk␣
            Reduction"/>
      <eLiterals name="EffortReduction" value="3" literal="Effort
            ␣Reduction"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EDataType" name="
          ReasoningObject" instanceClassName="org.eclipse.emf.
          common.util.Enumerator">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
        <details key="name" value="Reasoning:Object"/>
        <details key="baseType" value="Reasoning"/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Relationship">
```

```
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="name" value="Relationship"/>
        <details key="kind" value="empty"/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="customerEquity" eType="#//CustomerEquity">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="CustomerEquity"/>
        </eAnnotations>
      </eStructuralFeatures>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="RevenueModel">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="name" value="RevenueModel"/>
        <details key="kind" value="empty"/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="revenueModelID" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//ID" iD="true">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="RevenueModelID"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="Name"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="Description"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="streamType" eType="#//StreamType">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="StreamType"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="pricingMethod" eType="#//PricingMethod">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="PricingMethod"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="RM_TC" lowerBound="1" upperBound="-1"
        eType="#//TargetCustomer" resolveProxies="false">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="RM_TC"/>
      </eAnnotations>
    </eStructuralFeatures>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="StreamType">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="StreamType"/>
    </eAnnotations>
    <eLiterals name="NotDefined" literal="Not␣Defined"/>
    <eLiterals name="Selling" value="1"/>
    <eLiterals name="Lending" value="2"/>
    <eLiterals name="Licensing" value="3"/>
    <eLiterals name="TransactionCut" value="4" literal="Transaction␣Cut"/>
    <eLiterals name="Advertising" value="5"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="StreamTypeObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
```

```
      <details key="name" value="StreamType:Object"/>
      <details key="baseType" value="StreamType"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="TargetCustomer">
    <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
      <details key="name" value="TargetCustomer"/>
      <details key="kind" value="elementOnly"/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EReference" name="consistsOf" lowerBound="1"
        eType="#//Relationship" containment="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="element"/>
        <details key="name" value="consistsOf"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="targetCustomerID" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//ID" iD="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="TargetCustomerID"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Name"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Description"/>
      </eAnnotations>
      </eStructuralFeatures>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="ValueLevel">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
          ExtendedMetaData">
        <details key="name" value="ValueLevel"/>
      </eAnnotations>
      <eLiterals name="NotDefined" literal="Not␣Defined"/>
      <eLiterals name="Commodity" value="1"/>
      <eLiterals name="InnovativeImitation" value="2" literal="
          Innovative␣Imitation"/>
      <eLiterals name="Excellence" value="3"/>
      <eLiterals name="Innovation" value="4"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EDataType" name="
        ValueLevelObject" instanceClassName="org.eclipse.emf.
        common.util.Enumerator">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
          ExtendedMetaData">
        <details key="name" value="ValueLevel:Object"/>
        <details key="baseType" value="ValueLevel"/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="ValueModel">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/
          ExtendedMetaData">
        <details key="name" value="ValueModel"/>
        <details key="kind" value="elementOnly"/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
          RevenueModel" lowerBound="1"
          upperBound="-1" eType="#//RevenueModel" containment="
              true">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
          <details key="kind" value="element"/>
          <details key="name" value="RevenueModel"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
          TargetCustomer" lowerBound="1"
          upperBound="-1" eType="#//TargetCustomer" containment="
              true">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/
            ExtendedMetaData">
          <details key="kind" value="element"/>
        <details key="name" value="TargetCustomer"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ValueObject" lowerBound="1"
        upperBound="-1" eType="#//ValueObject" containment="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="element"/>
        <details key="name" value="ValueObject"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="DistributionChannel" lowerBound="1"
        upperBound="-1" eType="#//DistributionChannel" containment="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="element"/>
        <details key="name" value="DistributionChannel"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ValueOffer" lowerBound="1"
        eType="#//ValueOffer" containment="true">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="element"/>
        <details key="name" value="ValueOffer"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="author" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String"
        defaultValueLiteral="">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Author"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="created" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//Date">
```

```
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="Created"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="lastModified" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//Date">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="LastModified"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="Version"/>
        </eAnnotations>
      </eStructuralFeatures>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="ValueObject">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="name" value="ValueObject"/>
        <details key="kind" value="empty"/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="valueObjectID" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//ID" iD="true">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
          <details key="kind" value="attribute"/>
          <details key="name" value="ValueObjectID"/>
        </eAnnotations>
      </eStructuralFeatures>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
        <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
    <details key="kind" value="attribute"/>
    <details key="name" value="Name"/>
   </eAnnotations>
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
   <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
    <details key="kind" value="attribute"/>
    <details key="name" value="Description"/>
   </eAnnotations>
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" eType="#//ValueObjectType">
   <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
    <details key="kind" value="attribute"/>
    <details key="name" value="Type"/>
   </eAnnotations>
  </eStructuralFeatures>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="ValueObjectType">
  <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
   <details key="name" value="ValueObjectType"/>
  </eAnnotations>
  <eLiterals name="NotDefined" literal="Not␣Defined"/>
  <eLiterals name="Tangible" value="1"/>
  <eLiterals name="Intangible" value="2"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EDataType" name="ValueObjectTypeObject" instanceClassName="org.eclipse.emf.common.util.Enumerator">
  <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
   <details key="name" value="ValueObjectType:Object"/>
   <details key="baseType" value="ValueObjectType"/>
  </eAnnotations>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="ValueOffer">
  <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
   <details key="name" value="ValueOffer"/>
   <details key="kind" value="empty"/>
  </eAnnotations>
```

```
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Name"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType␣http://www.eclipse.org/emf/2003/XMLType#//String">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Description"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="reasoning" eType="#//Reasoning">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="Reasoning"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="valueLevel" eType="#//ValueLevel">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="ValueLevel"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="priceLevel" eType="#//PriceLevel">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="PriceLevel"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="lifeCycleStep" eType="#//LifeCycleStep">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
```

```
        <details key="kind" value="attribute"/>
        <details key="name" value="LifeCycleStep"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="VOf_DC" lowerBound="1"
        upperBound="-1" eType="#//DistributionChannel" resolveProxies="false">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="VOf_DC"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="VOf_RM" lowerBound="1"
        upperBound="-1" eType="#//RevenueModel" resolveProxies="false">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="VOf_RM"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="VOf_TC" lowerBound="1"
        upperBound="-1" eType="#//TargetCustomer" resolveProxies="false">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="VOf_TC"/>
      </eAnnotations>
    </eStructuralFeatures>
    <eStructuralFeatures xsi:type="ecore:EReference" name="VOf_VOb" lowerBound="1"
        upperBound="-1" eType="#//ValueObject" resolveProxies="false">
      <eAnnotations source="http:///org/eclipse/emf/ecore/util/ExtendedMetaData">
        <details key="kind" value="attribute"/>
        <details key="name" value="VOf_VOb"/>
      </eAnnotations>
    </eStructuralFeatures>
  </eClassifiers>
</ecore:EPackage>
```

Listing A.2: CSMM Ecore

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="CSM"
    nsURI="http://www.siemens.com/ct/texo/ise" nsPrefix="csm">
  <eClassifiers xsi:type="ecore:EClass" name="CSM">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="CSM_SProduct" lowerBound="1"
        upperBound="-1" eType="#//ServiceProduct" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ServiceProduct">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="key" lowerBound="1" eType="#//URI"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="documentation" lowerBound="1"
        eType="#//URI"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="spokenLanguage" lowerBound="1"
        upperBound="-1" eType="#//Lang"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="writtenLanguage" lowerBound="1"
        upperBound="-1" eType="#//Lang"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="created" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="updated" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="nextUpdate" upperBound="-1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType="#//ServiceType"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="automation" lowerBound="1"
        eType="#//AutomationLevel"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="composition" lowerBound="1"
        eType="#//CompositionLevel"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="customizable" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SP_ref_SP" upperBound="-1"
        eType="#//ServiceProduct"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Payment" upperBound="-1"
        eType="#//Payment" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Discount" upperBound="-1"
        eType="#//Discount" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_TReport" upperBound="-1"
        eType="#//TestReport" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Certificate" upperBound="-1"
        eType="#//Certificate" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Rating" upperBound="-1"
        eType="#//Rating" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Standard" upperBound="-1"
        eType="#//Standard" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SProduct_Classification"
```

```
        upperBound="-1" eType="#//Classification" containment="
            true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_TermsOfUse" eType="#//TermsOfUse"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Benefit" upperBound="-1"
        eType="#//Benefit" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Right" upperBound="-1"
        eType="#//Right" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Actor" upperBound="-1"
        eType="#//Actor" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Resource" upperBound="-1"
        eType="#//Resource" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Quality" upperBound="-1"
        eType="#//Quality" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Price" upperBound="-1"
        eType="#//Price" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Channel" upperBound="-1"
        eType="#//Channel" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        SProduct_Capability" lowerBound="1"
        upperBound="-1" eType="#//Capability" containment="true
            "/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ServiceType">
    <eLiterals name="CoreService" literal="Core␣Service"/>
    <eLiterals name="SupportingService" value="1" literal="
        Supporting␣Service"/>
    <eLiterals name="EnhancingService" value="2" literal="
        Enhancing␣Service"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="AutomationLevel">
    <eLiterals name="FullyAutomated" literal="Fully␣Automated"/
        >
    <eLiterals name="PartiallyAutomated" value="1" literal="
        Partially␣Automated"/>
    <eLiterals name="Manual" value="2"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="
```

```
          ClassificationSystem">
      <eLiterals name="NAICS"/>
      <eLiterals name="eClass" value="1" literal="eCl@ss"/>
      <eLiterals name="UNSPSC" value="2"/>
      <eLiterals name="NiceClassification" value="3" literal="
          Nice␣Classification"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="ResourceType">
      <eLiterals name="PhysicalGood" literal="Physical␣Good"/>
      <eLiterals name="Service" value="1"/>
      <eLiterals name="Information" value="2"/>
      <eLiterals name="Media" value="3"/>
      <eLiterals name="Personnel" value="4"/>
      <eLiterals name="Capability" value="5"/>
      <eLiterals name="Experience" value="6"/>
      <eLiterals name="Monetary" value="7"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="RightType">
      <eLiterals name="Copyright"/>
      <eLiterals name="PropertyRight" value="1" literal="Property
          ␣Right"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="Lang">
      <eLiterals name="eng"/>
      <eLiterals name="ger" value="1"/>
      <eLiterals name="fra" value="2"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="Currency">
      <eLiterals name="EUR"/>
      <eLiterals name="USD" value="1"/>
      <eLiterals name="AUD" value="2"/>
      <eLiterals name="JPY" value="3"/>
      <eLiterals name="Others" value="4"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="TimeGranularity">
      <eLiterals name="Year"/>
      <eLiterals name="Month" value="1"/>
      <eLiterals name="Week" value="2"/>
      <eLiterals name="Day" value="3"/>
      <eLiterals name="Hour" value="4"/>
      <eLiterals name="Minute" value="5"/>
      <eLiterals name="Second" value="6"/>
      <eLiterals name="Millisecond" value="7"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="PaymentInstrument"
        >
      <eLiterals name="Cash"/>
      <eLiterals name="VISA" value="1"/>
      <eLiterals name="Mastercard" value="2"/>
      <eLiterals name="AmEx" value="3"/>
      <eLiterals name="DebitCard" value="4" literal="Debit␣Card"/
          >
      <eLiterals name="Token" value="5"/>
      <eLiterals name="Cheque" value="6"/>
      <eLiterals name="Coupon" value="7"/>
      <eLiterals name="Voucher" value="8"/>
      <eLiterals name="BankTransfer" value="9" literal="Bank␣
          Transfer"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="Servqual">
      <eLiterals name="Tangibles"/>
      <eLiterals name="Reliability" value="1"/>
      <eLiterals name="Responsiveness" value="2"/>
      <eLiterals name="Assurance" value="3"/>
      <eLiterals name="Empathy" value="4"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="EightPs">
      <eLiterals name="People"/>
      <eLiterals name="PhysicalEvidence" value="1" literal="
          Physical␣Evidence"/>
      <eLiterals name="Place" value="2"/>
      <eLiterals name="Price" value="3"/>
      <eLiterals name="Process" value="4"/>
      <eLiterals name="Product" value="5"/>
      <eLiterals name="ProductivityAndQuality" value="6" literal=
          "Productivity␣&amp;␣Quality"/>
      <eLiterals name="Promotion" value="7"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="PriceModifier">
      <eLiterals name="Exact"/>
      <eLiterals name="LimitedTo" value="1" literal="Limited␣To"/
          >
      <eLiterals name="StartingFrom" value="2" literal="Starting␣
          From"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="InterfaceType">
      <eLiterals name="PersonellInterface" literal="Personell␣
          Interface"/>
      <eLiterals name="WebInterface" value="1" literal="Web␣
          Interface"/>
      <eLiterals name="TechnicalInterface" value="2" literal="
          Technical␣Interface"/>
```

```
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="Recurrence">
    <eLiterals name="Yearly"/>
    <eLiterals name="Monthly" value="1"/>
    <eLiterals name="Weekly" value="2"/>
    <eLiterals name="Dayly" value="3"/>
    <eLiterals name="Hourly" value="4"/>
    <eLiterals name="EveryMinute" value="5" literal="Every␣
        Minute"/>
    <eLiterals name="EverySecond" value="6" literal="Every␣
        Second"/>
    <eLiterals name="EveryMillisecond" value="7" literal="Every
        ␣Millisecond"/>
    <eLiterals name="once" value="8"/>
    <eLiterals name="never" value="9"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="CompositionLevel">
    <eLiterals name="IntermediaryService" literal="Intermediary
        ␣Service"/>
    <eLiterals name="FinalService" value="1" literal="Final␣
        Service"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ChannelType">
    <eLiterals name="Physically"/>
    <eLiterals name="Electronically" value="1"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="URI"
      instanceClassName="java.net.URI"/>
  <eClassifiers xsi:type="ecore:EClass" name="Payment">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
        " lowerBound="1" eType="ecore:EDataType␣http://www.
        eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        instrument" lowerBound="1"
        eType="#//PaymentInstrument"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        preferred" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        recurrence" lowerBound="1"
        eType="#//Recurrence"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
          Payment_ref_Actor" eType="#//Actor"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Discount"
        abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
          " lowerBound="1" eType="ecore:EDataType␣http://www.
          eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          allowance" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EDouble"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="TestReport">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          provider" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          title" lowerBound="1" eType="ecore:EDataType␣http://www
          .eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          reference" lowerBound="1"
        eType="#//URI"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          created" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EDate"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Certificate">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          provider" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          title" lowerBound="1" eType="ecore:EDataType␣http://www
          .eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          created" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EDate"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Rating">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
          " lowerBound="1" eType="ecore:EDataType␣http://www.
          eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="created" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="comment" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="servqual" upperBound="-1"
        eType="#//Servqual"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="EightPs" upperBound="-1"
        eType="#//EightPs"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Rating_ref_Consumer" eType="#//Consumer"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Standard">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="provider" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="title" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="status" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="created" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="author" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Classification">
```

```
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           value" lowerBound="1" eType="ecore:EDataType␣http://www
           .eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           system" lowerBound="1"
           eType="#//ClassificationSystem"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="TermsOfUse">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           paymentCondition" lowerBound="1"
           eType="#//URI"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           deliveryCondition" lowerBound="1"
           eType="#//URI"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Benefit">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
           " lowerBound="1" eType="ecore:EDataType␣http://www.
           eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           description" lowerBound="1"
           eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
               Ecore#//EString"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Right">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           description" lowerBound="1"
           eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
               Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
           " lowerBound="1" eType="ecore:EDataType␣http://www.
           eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="type
           " lowerBound="1" eType="#//RightType"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           Right_ref_Resource" lowerBound="1"
           upperBound="-1" eType="#//Resource"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="PaymentDiscount"
         eSuperTypes="#//Discount">
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           PDiscount_ref_Payment"
           lowerBound="1" upperBound="-1" eType="#//Payment"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="SeasonalDiscount"
          eSuperTypes="#//Discount">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="from" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="to" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDate"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Actor" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="key" lowerBound="1" eType="#//URI"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="DUNS" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="signature" lowerBound="1"
        eType="#//XMLSignature"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="industry" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Actor_ref_Right" upperBound="-1"
        eType="#//Right"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Actor_Contact" upperBound="-1"
        eType="#//Contact" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="XMLSignature" instanceClassName="javax.xml.crypto.dsig.XMLSignature"/>
  <eClassifiers xsi:type="ecore:EClass" name="Provider" eSuperTypes="#//Actor"/>
  <eClassifiers xsi:type="ecore:EClass" name="Consumer" eSuperTypes="#//Actor"/>
  <eClassifiers xsi:type="ecore:EClass" name="Partner" eSuperTypes="#//Actor"/>
  <eClassifiers xsi:type="ecore:EClass" name="Contact">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="personName" lowerBound="1"
```

```
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="phone" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="email" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Contact_ALine" lowerBound="1"
        upperBound="-1" eType="#//AddressLine" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="AddressLine">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="keyName" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="keyValue" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Resource">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType="#//ResourceType"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Quality">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Quality_Performance" eType="#//Performance"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
```

```
          Quality_Dependability"
          eType="#//Dependability" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Quality_Security" eType="#//Security"
          containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Performance">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="capacity" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Performance_Latency" eType="#//Latency"
          containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Performance_Throughput"
          eType="#//Throughput" containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Dependability">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="availability" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="reliability" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="reliabilityGran" lowerBound="1"
          eType="#//TimeGranularity"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="maintainability" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="maintainaGran" lowerBound="1"
          eType="#//TimeGranularity"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="accuracy" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Security">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="authentication" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
          Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="authorization" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Security_Confidentiality"
        eType="#//Confidentiality" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Security_DIntegrity" eType="#//DataIntegrity"
        containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Latency">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="granularity" lowerBound="1"
        eType="#//TimeGranularity"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Throughput">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="events" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="recurrence" lowerBound="1"
        eType="#//Recurrence"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="DataIntegrity">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="granularity" lowerBound="1"
        eType="#//TimeGranularity"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Confidentiality">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="encrypted" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="keyLength" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="encryptType" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Condition" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="state" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Condition_ref_Resource"
        lowerBound="1" eType="#//Resource"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="PreCondition" eSuperTypes="#//Condition"/>
  <eClassifiers xsi:type="ecore:EClass" name="Channel">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="channelLength" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="productVariety" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="waitingTime" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="waitingTimeGran" lowerBound="1"
        eType="#//TimeGranularity"/>
```

```
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType="#//ChannelType"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Channel_ref_Actor" eType="#//Actor"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Price" abstract="true">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Pricing" abstract="true" eSuperTypes="#//Price">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="amount" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="currency" lowerBound="1"
          eType="#//Currency"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="tax" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="taxInclusive" lowerBound="1"
          eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="modifier" lowerBound="1"
          eType="#//PriceModifier"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="TransactionCut" eSuperTypes="#//Price">
      <eStructuralFeatures xsi:type="ecore:EReference" name="TCut_ref_Actor" eType="#//Actor"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Flatrate" eSuperTypes="#//Pricing">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="recurrence" lowerBound="1"
          eType="#//Recurrence"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="UsageBased" eSuperTypes="#//Pricing"/>
    <eClassifiers xsi:type="ecore:EClass" name="TwoPartTariff" eSuperTypes="#//Pricing">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="fixedSum" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="recurrence" lowerBound="1"
        eType="#//Recurrence"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="NBlockTariff" eSuperTypes="#//Pricing">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="n" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="allowancePoints" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="PostCondition" eSuperTypes="#//Condition"/>
  <eClassifiers xsi:type="ecore:EClass" name="Capability">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="interface" lowerBound="1"
        eType="#//InterfaceType"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="duration" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="durationGran" lowerBound="1"
        eType="#//TimeGranularity"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Capability_ref_Price" upperBound="-1"
        eType="#//Price"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Capability_Condition" upperBound="-1"
        eType="#//Condition" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Capability_ref_Quality"
```

```
        eType="#//Quality"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ChannelDiscount"
      eSuperTypes="#//Discount">
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        CDiscount_ref_Channel"
        lowerBound="1" upperBound="-1" eType="#//Channel"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="
      QuantitativeDiscount" eSuperTypes="#//Discount">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        quantity" lowerBound="1"
        eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
            Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
        QDiscount_ref_Cap" lowerBound="1"
        upperBound="-1" eType="#//Capability"/>
  </eClassifiers>
</ecore:EPackage>
```

Listing A.3: WSDL Ecore

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3
        .org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="
        wsdl"
    nsURI="http://www.eclipse.org/wsdl/2003/WSDL" nsPrefix="
        wsdl">
  <eAnnotations source="http://www.eclipse.org/emf/2002/
      GenModel">
    <details key="documentation" value="The␣WSDL␣model␣contains
        ␣classes␣for␣the␣Web␣Services␣Description␣Language␣(
        WSDL).&#xD;&#xA;&#xD;&#xA;WSDL␣describes␣network␣
        services␣as␣sets␣of␣endpoints␣operating␣on␣messages.␣
        The␣operations␣and␣messages␣are␣described␣abstractly,␣
        and␣then␣bound␣to␣a␣concrete␣network␣protocol␣and␣
        message␣format␣to␣define␣an␣endpoint.&#xD;&#xA;&#xD;&#
        xA;WSDL␣describes␣the␣formats␣of␣the␣messages␣exchanged
        ␣by␣the␣services,␣and␣supports␣the␣XML␣Schemas␣
        specification␣as␣its␣canonical␣type␣system.␣This␣
        package␣uses␣an␣XML␣Schema␣Infoset␣model␣package␣(see␣
        the␣XSD␣package)␣to␣describe␣the␣abstract␣message␣
        formats.&#xD;&#xA;&#xD;&#xA;The␣model␣contains␣the␣
        following␣diagrams,␣named␣after␣the␣corresponding␣
```

```
          chapters␣in␣the␣WSDL␣1.1␣specification␣(http://www.w3.
          org/TR/2001/NOTE-wsdl-20010315)&#xD;&#xA;-␣2.1␣
          Definition,␣shows␣the␣WSDL␣definition␣element␣and␣the␣
          WSDL␣document␣structure&#xD;&#xA;-␣2.1.1␣Naming␣and␣
          Linking,␣shows␣the␣namespace␣and␣import␣mechanism&#xD
          ;&#xA;-␣2.1.3␣Extensibility,␣shows␣the␣WSDL␣
          extensibility␣mechanism&#xD;&#xA;-␣2.2␣Types,␣shows␣the
          ␣use␣of␣XML␣Schema␣types␣in␣WSDL&#xD;&#xA;-␣2.3␣
          Messages,␣2.4␣PortTypes,␣2.5␣Bindings␣and␣2.7␣Services,
          ␣show␣the␣major␣WSDL␣elements␣and␣their␣relations.&#xD
          ;&#xA;&#xD;&#xA;The␣WSDL␣classes␣extend␣the␣javax.wsdl␣
          interfaces␣defined␣by␣JSR␣110.␣Classes␣with␣interface␣
          and␣datatype␣stereotypes␣are␣used␣to␣represent␣these␣
          non-MOF␣interfaces."/>
  </eAnnotations>
  <eClassifiers xsi:type="ecore:EClass" name="WSDLElement"
      abstract="true">
    <eAnnotations source="http://www.eclipse.org/emf/2002/
        GenModel">
      <details key="documentation" value="This␣class␣represents
          ␣a␣WSDL␣language␣element."/>
    </eAnnotations>
    <eOperations name="getEnclosingDefinition" eType="#//
        Definition"/>
    <eOperations name="setEnclosingDefinition">
      <eParameters name="definition" eType="#//Definition"/>
    </eOperations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        documentationElement" eType="#//DOMElement"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        element" eType="#//DOMElement"
        transient="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="PortType"
      eSuperTypes="#//ExtensibleElement␣#//IPortType">
    <eAnnotations source="http://www.eclipse.org/emf/2002/
        GenModel">
      <details key="documentation" value="This␣class␣represents
          ␣a␣WSDL␣portType␣element␣of␣the␣WSDL␣specification␣
          version␣1.1␣and␣an␣Interface␣component␣of␣the␣WSDL␣
          specification␣version␣1.2.␣A␣port␣type␣or␣Interface␣
          is␣a␣named␣set␣of␣abstract␣operations␣and␣the␣
          abstract␣messages␣involved."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
        qName" eType="#//QName"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="undefined" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eOperations" upperBound="-1"
        eType="#//Operation" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Operation" eSuperTypes="#//ExtensibleElement␣#//IOperation">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣operation␣element.␣A␣WSDL␣operation␣is␣an␣abstract␣description␣of␣an␣action␣supported␣by␣a␣service."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="style" eType="#//OperationType"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="undefined" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eInput" eType="#//Input"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eOutput" eType="#//Output"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eFaults" upperBound="-1"
        eType="#//Fault" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eParameterOrdering" upperBound="-1"
        eType="#//Part"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Message" eSuperTypes="#//ExtensibleElement␣#//IMessage">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣message␣element.␣A␣WSDL␣message␣is␣an␣abstract,␣typed␣definition␣of␣the␣data␣being␣communicated."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="qName" eType="#//QName"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="undefined" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eParts" upperBound="-1"
        eType="#//Part" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Part" eSuperTypes="#//ExtensibleElement␣#//IPart">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣part␣element.␣Parts␣describe␣the␣logical␣abstract␣content␣of␣a␣message.␣Each␣part␣is␣associated␣with␣a␣type␣from␣some␣type␣system.␣"/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="elementName" eType="#//QName"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="typeName" eType="#//QName"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="typeDefinition" eType="ecore:EClass␣platform:/plugin/org.eclipse.xsd/model/XSD.ecore#//XSDTypeDefinition"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="elementDeclaration" eType="ecore:EClass␣platform:/plugin/org.eclipse.xsd/model/XSD.ecore#//XSDElementDeclaration"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eMessage" eType="#//Message"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Binding" eSuperTypes="#//ExtensibleElement␣#//IBinding">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣binding␣element.␣A␣binding␣defines␣message␣format␣and␣protocol␣details␣for␣operations␣and␣messages␣defined␣by␣a␣particular␣portType.␣There␣may␣be␣any␣number␣of␣bindings␣for␣a␣given␣portType."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
```

```
           qName" eType="#//QName"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
           undefined" eType="ecore:EDataType␣http://www.eclipse.
           org/emf/2002/Ecore#//EBoolean"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           ePortType" lowerBound="1"
          eType="#//PortType"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           eBindingOperations" upperBound="-1"
          eType="#//BindingOperation" containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="BindingOperation"
          eSuperTypes="#//ExtensibleElement␣#//IBindingOperation">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
           GenModel">
        <details key="documentation" value="This␣class␣represents
             ␣a␣WSDL␣operation␣element␣within␣a␣binding.␣An␣
             operation␣element␣within␣a␣binding␣specifies␣binding␣
             information␣for␣the␣operation␣with␣the␣same␣name␣
             within␣the␣binding's␣portType.␣"/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
           " eType="ecore:EDataType␣http://www.eclipse.org/emf
           /2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           eOperation" lowerBound="1"
          eType="#//Operation"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           eBindingInput" eType="#//BindingInput"
          containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           eBindingOutput" eType="#//BindingOutput"
          containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
           eBindingFaults" upperBound="-1"
          eType="#//BindingFault" containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Service"
         eSuperTypes="#//ExtensibleElement␣#//IService">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
           GenModel">
        <details key="documentation" value="This␣class␣represents
             ␣a␣WSDL␣service␣element.␣A␣service␣groups␣a␣set␣of␣
             related␣ports␣together."/>
      </eAnnotations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
```

```
qName" eType="#//QName"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="undefined" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ePorts" upperBound="-1"
        eType="#//Port" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Port" eSuperTypes="#//ExtensibleElement␣#//IPort">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣port␣element.␣A␣port␣defines␣an␣individual␣endpoint␣by␣specifying␣a␣single␣address␣for␣a␣binding"/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eBinding" lowerBound="1"
        eType="#//Binding"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ExtensibilityElement" eSuperTypes="#//WSDLElement␣#//IExtensibilityElement">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣extensibility␣element.␣WSDL␣allows␣extensibility␣elements␣representing␣a␣specific␣technology␣under␣various␣elements␣defined␣by␣WSDL."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="required" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="elementType" eType="#//QName"
        transient="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Definition" eSuperTypes="#//ExtensibleElement␣#//IDefinition">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents
```

```
                ␣a␣WSDL␣definitions␣element.␣The␣WSDL␣definitions␣
                element␣is␣the␣root␣element␣of␣a␣WSDL␣document."/>
      </eAnnotations>
      <eOperations name="getDocument" eType="#//DOMDocument"/>
      <eOperations name="setDocument">
        <eParameters name="document" eType="#//DOMDocument"/>
      </eOperations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            targetNamespace" eType="ecore:EDataType␣http://www.
            eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            location" eType="ecore:EDataType␣http://www.eclipse.org
            /emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            qName" eType="#//QName"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            encoding" eType="ecore:EDataType␣http://www.eclipse.org
            /emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eImports" upperBound="-1"
          eType="#//Import" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eTypes" eType="#//Types"
          containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eMessages" upperBound="-1"
          eType="#//Message" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            ePortTypes" upperBound="-1"
          eType="#//PortType" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eBindings" upperBound="-1"
          eType="#//Binding" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eServices" upperBound="-1"
          eType="#//Service" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            eNamespaces" upperBound="-1"
          eType="#//Namespace" containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Import"
          eSuperTypes="#//ExtensibleElement␣#//IImport">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
        <details key="documentation" value="This␣class␣represents
              ␣WSDL␣import␣element.␣WSDL␣allows␣associating␣a␣
              namespace␣with␣a␣document␣location␣using␣an␣import␣
              element."/>
        </eAnnotations>
        <eOperations name="getSchema" eType="ecore:EClass␣platform:
            /plugin/org.eclipse.xsd/model/XSD.ecore#//XSDSchema"/>
        <eOperations name="setSchema">
          <eParameters name="schema" eType="ecore:EClass␣platform:/
              plugin/org.eclipse.xsd/model/XSD.ecore#//XSDSchema"/>
        </eOperations>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            namespaceURI" eType="ecore:EDataType␣http://www.eclipse
            .org/emf/2002/Ecore#//EString"/>
        <eStructuralFeatures xsi:type="ecore:EAttribute" name="
            locationURI" eType="ecore:EDataType␣http://www.eclipse.
            org/emf/2002/Ecore#//EString"/>
        <eStructuralFeatures xsi:type="ecore:EReference" name="
            eDefinition" eType="#//Definition"/>
        <eStructuralFeatures xsi:type="ecore:EReference" name="
            eSchema" eType="ecore:EClass␣platform:/plugin/org.
            eclipse.xsd/model/XSD.ecore#//XSDSchema"/>
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="ExtensibleElement
          " abstract="true" eSuperTypes="#//WSDLElement␣#//
          IElementExtensible␣#//IAttributeExtensible">
        <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
          <details key="documentation" value="␣WSDL␣allows␣elements
              ␣representing␣a␣specific␣technology␣(referred␣to␣here
              ␣as␣extensibility␣elements)␣under␣various␣elements␣
              defined␣by␣WSDL.␣This␣class␣represents␣a␣WSDL␣point␣
              of␣extensibility."/>
        </eAnnotations>
        <eStructuralFeatures xsi:type="ecore:EReference" name="
            eExtensibilityElements"
            upperBound="-1" eType="#//ExtensibilityElement"
                containment="true"/>
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="Input"
          eSuperTypes="#//MessageReference␣#//IInput">
        <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
          <details key="documentation" value="This␣class␣represents
              ␣a␣WSDL␣input␣element.␣An␣input␣element␣specifies␣the
              ␣abstract␣message␣format␣for␣the␣input␣of␣the␣
              operation."/>
        </eAnnotations>
```

```
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Output"
        eSuperTypes="#//MessageReference␣#//IOutput">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣a␣WSDL␣output␣element.␣An␣output␣element␣specifies␣
            the␣abstract␣message␣format␣for␣the␣output␣of␣the␣
            operation."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Fault"
        eSuperTypes="#//MessageReference␣#//IFault">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣a␣WSDL␣fault␣element.␣Fault␣elements␣specify␣the␣
            abstract␣message␣format␣for␣any␣error␣messages␣that␣
            may␣be␣output␣as␣the␣result␣of␣the␣operation␣"/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="BindingInput"
        eSuperTypes="#//ExtensibleElement␣#//IBindingInput">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣a␣WSDL␣input␣element␣within␣a␣operation␣within␣a␣
            binding.␣An␣input␣element␣within␣an␣operation␣within␣
            a␣binding␣specifies␣binding␣information␣for␣the␣input
            ␣of␣the␣operation.␣"/>
      </eAnnotations>
      <eOperations name="getInput" eType="#//IInput"/>
      <eOperations name="setInput">
        <eParameters name="input" eType="#//IInput"/>
      </eOperations>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
          " eType="ecore:EDataType␣http://www.eclipse.org/emf
          /2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="
          eInput" lowerBound="1"
         eType="#//Input"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="BindingOutput"
        eSuperTypes="#//ExtensibleElement␣#//IBindingOutput">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣output␣element␣within␣a␣operation␣within␣a␣binding.␣An␣output␣element␣within␣an␣operation␣within␣a␣binding␣specifies␣binding␣information␣for␣the␣output␣of␣the␣operation.␣"/>
    </eAnnotations>
    <eOperations name="getOutput" eType="#//IOutput"/>
    <eOperations name="setOutput">
      <eParameters name="output" eType="#//IOutput"/>
    </eOperations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eOutput" lowerBound="1"
        eType="#//Output"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="BindingFault" eSuperTypes="#//ExtensibleElement␣#//IBindingFault">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣WSDL␣fault␣element␣within␣a␣operation␣within␣a␣binding.␣A␣fault␣element␣within␣an␣operation␣within␣a␣binding␣specifies␣binding␣information␣for␣the␣fault␣with␣the␣same␣name.␣"/>
    </eAnnotations>
    <eOperations name="getFault" eType="#//IFault"/>
    <eOperations name="setFault">
      <eParameters name="fault" eType="#//IFault"/>
    </eOperations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="eFault" lowerBound="1"
        eType="#//Fault"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="QName" instanceClassName="javax.xml.namespace.QName">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.QName␣class.␣A␣QName␣is␣a␣fully␣qualified␣name."/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Namespace">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣a␣namespace␣and␣the␣corresponding␣namespace␣prefix␣used␣in␣a␣WSDL␣document."/>
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="URI" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="prefix" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="OperationType" instanceClassName="javax.wsdl.OperationType">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="ONE_WAY=1&#xD;&#xA;REQUEST_RESPONSE=2&#xD;&#xA;SOLICIT_RESPONSE=3&#xD;&#xA;NOTIFICATION=4"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IPortType" instanceClassName="javax.wsdl.PortType"
      abstract="true" interface="true" eSuperTypes="#//IAttributeExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.PortType␣non-MOF␣interface."/>
    </eAnnotations>
    <eOperations name="addOperation">
      <eParameters name="operation" eType="#//IOperation"/>
    </eOperations>
    <eOperations name="getOperation" eType="#//IOperation">
      <eParameters name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      <eParameters name="inputName" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      <eParameters name="outputName" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="getOperations" eType="#//IList"/>
  </eClassifiers>
```

```
    <eClassifiers xsi:type="ecore:EClass" name="IOperation"
         instanceClassName="javax.wsdl.Operation"
        abstract="true" interface="true" eSuperTypes="#//
            IElementExtensible">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣javax.wsdl.Operation␣non-MOF␣interface."/>
      </eAnnotations>
      <eOperations name="addFault">
        <eParameters name="fault" eType="#//IFault"/>
      </eOperations>
      <eOperations name="getFault" eType="#//IFault">
        <eParameters name="name" eType="ecore:EDataType␣http://
            www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eOperations>
      <eOperations name="getFaults" eType="#//IMap"/>
      <eOperations name="getParameterOrdering" eType="#//IList"/>
      <eOperations name="setParameterOrdering">
        <eParameters name="parameterOrder" eType="#//IList"/>
      </eOperations>
      <eOperations name="getInput" eType="#//IInput"/>
      <eOperations name="setInput">
        <eParameters name="input" eType="#//IInput"/>
      </eOperations>
      <eOperations name="getOutput" eType="#//IOutput"/>
      <eOperations name="setOutput">
        <eParameters name="output" eType="#//IOutput"/>
      </eOperations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IInput"
         instanceClassName="javax.wsdl.Input"
        abstract="true" interface="true" eSuperTypes="#//
            IAttributeExtensible">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣javax.wsdl.Input␣non-MOF␣interface."/>
      </eAnnotations>
      <eOperations name="getMessage" eType="#//IMessage"/>
      <eOperations name="setMessage">
        <eParameters name="message" eType="#//IMessage"/>
      </eOperations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IOutput"
         instanceClassName="javax.wsdl.Output"
      abstract="true" interface="true" eSuperTypes="#//IAttributeExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Output␣non-MOF␣interface."/>
    </eAnnotations>
    <eOperations name="getMessage" eType="#//IMessage"/>
    <eOperations name="setMessage">
      <eParameters name="message" eType="#//IMessage"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IFault" instanceClassName="javax.wsdl.Fault"
      abstract="true" interface="true" eSuperTypes="#//IAttributeExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Fault␣non-MOF␣interface."/>
    </eAnnotations>
    <eOperations name="getMessage" eType="#//IMessage"/>
    <eOperations name="setMessage">
      <eParameters name="message" eType="#//IMessage"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IMessage" instanceClassName="javax.wsdl.Message"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Message␣non-MOF␣interface"/>
    </eAnnotations>
    <eOperations name="addPart">
      <eParameters name="part" eType="#//IPart"/>
    </eOperations>
    <eOperations name="getPart" eType="#//IPart">
      <eParameters name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="getParts" eType="#//IMap"/>
    <eOperations name="getOrderedParts" eType="#//IList">
      <eParameters name="partOrder" eType="#//IList"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IPart" instanceClassName="javax.wsdl.Part"
      abstract="true" interface="true" eSuperTypes="#//IAttributeExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Part␣non-MOF␣interface"/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IService" instanceClassName="javax.wsdl.Service"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Service␣non-MOF␣interface."/>
    </eAnnotations>
    <eOperations name="addPort">
      <eParameters name="port" eType="#//IPort"/>
    </eOperations>
    <eOperations name="getPorts" eType="#//IMap"/>
    <eOperations name="getPort" eType="#//IPort">
      <eParameters name="name" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IPort" instanceClassName="javax.wsdl.Port"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Port␣non-MOF␣interface."/>
    </eAnnotations>
    <eOperations name="getBinding" eType="#//IBinding"/>
    <eOperations name="setBinding">
      <eParameters name="binding" eType="#//IBinding"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IBinding" instanceClassName="javax.wsdl.Binding"
      abstract="true" interface="true" eSuperTypes="#//
```

```
        IElementExtensible">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣javax.wsdl.Binding␣non-MOF␣interface."/>
      </eAnnotations>
      <eOperations name="addBindingOperation">
        <eParameters name="bindingOperation" eType="#//
            IBindingOperation"/>
      </eOperations>
      <eOperations name="getBindingOperation" eType="#//
          IBindingOperation">
        <eParameters name="name" eType="ecore:EDataType␣http://
            www.eclipse.org/emf/2002/Ecore#//EString"/>
        <eParameters name="inputName" eType="ecore:EDataType␣
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
        <eParameters name="outputName" eType="ecore:EDataType␣
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eOperations>
      <eOperations name="getBindingOperations" eType="#//IList"/>
      <eOperations name="getPortType" eType="#//IPortType"/>
      <eOperations name="setPortType">
        <eParameters name="portType" eType="#//IPortType"/>
      </eOperations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IBindingOperation
        " instanceClassName="javax.wsdl.BindingOperation"
        abstract="true" interface="true" eSuperTypes="#//
            IElementExtensible">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣javax.wsdl.BindingOperation␣non-MOF␣interface."/
            >
      </eAnnotations>
      <eOperations name="addBindingFault">
        <eParameters name="bindingFault" eType="#//IBindingFault"
            />
      </eOperations>
      <eOperations name="getBindingFault" eType="#//IBindingFault
          ">
        <eParameters name="name" eType="ecore:EDataType␣http://
            www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eOperations>
      <eOperations name="getBindingFaults" eType="#//IMap"/>
      <eOperations name="getOperation" eType="#//IOperation"/>
    <eOperations name="setOperation">
      <eParameters name="operation" eType="#//IOperation"/>
    </eOperations>
    <eOperations name="getBindingInput" eType="#//IBindingInput"/>
    <eOperations name="setBindingInput">
      <eParameters name="bindingInput" eType="#//IBindingInput"/>
    </eOperations>
    <eOperations name="getBindingOutput" eType="#//IBindingOutput"/>
    <eOperations name="setBindingOutput">
      <eParameters name="bindingOutput" eType="#//IBindingOutput"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IBindingInput" instanceClassName="javax.wsdl.BindingInput"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.BindingInput␣non-MOF␣interface."/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IBindingOutput" instanceClassName="javax.wsdl.BindingOutput"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.BindingOutput␣non-MOF␣interface."/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IBindingFault" instanceClassName="javax.wsdl.BindingFault"
      abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
      <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.BindingFault␣non-MOF␣interface."/>
    </eAnnotations>
  </eClassifiers>
```

```
    <eClassifiers xsi:type="ecore:EDataType" name="DOMElement" instanceClassName="org.w3c.dom.Element">
      <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
        <details key="documentation" value="This␣class␣represents␣the␣org.w3c.dom.Element␣non-MOF␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IExtensibilityElement" instanceClassName="javax.wsdl.extensions.ExtensibilityElement"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
        <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.ExtensibilityElement␣non-MOF␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IDefinition" instanceClassName="javax.wsdl.Definition"
        abstract="true" interface="true" eSuperTypes="#//IElementExtensible">
      <eAnnotations source="http://www.eclipse.org/emf/2002/GenModel">
        <details key="documentation" value="This␣class␣represents␣the␣javax.wsdl.Definition␣non-MOF␣interface."/>
      </eAnnotations>
      <eOperations name="addBinding">
        <eParameters name="binding" eType="#//IBinding"/>
      </eOperations>
      <eOperations name="addImport">
        <eParameters name="importDef" eType="#//IImport"/>
      </eOperations>
      <eOperations name="addMessage">
        <eParameters name="message" eType="#//IMessage"/>
      </eOperations>
      <eOperations name="addNamespace">
        <eParameters name="prefix" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
        <eParameters name="namespaceURI" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eOperations>
      <eOperations name="addPortType">
        <eParameters name="portType" eType="#//IPortType"/>
      </eOperations>
    <eOperations name="addService">
      <eParameters name="service" eType="#//IService"/>
    </eOperations>
    <eOperations name="createBindingFault" eType="#//IBindingFault"/>
    <eOperations name="createBindingInput" eType="#//IBindingInput"/>
    <eOperations name="createBindingOutput" eType="#//IBindingOutput"/>
    <eOperations name="createBindingOperation" eType="#//IBindingOperation"/>
    <eOperations name="createBinding" eType="#//IBinding"/>
    <eOperations name="createFault" eType="#//IFault"/>
    <eOperations name="createImport" eType="#//IImport"/>
    <eOperations name="createInput" eType="#//IInput"/>
    <eOperations name="createMessage" eType="#//IMessage"/>
    <eOperations name="createOperation" eType="#//IOperation"/>
    <eOperations name="createOutput" eType="#//IOutput"/>
    <eOperations name="createPart" eType="#//IPart"/>
    <eOperations name="createPort" eType="#//IPort"/>
    <eOperations name="createPortType" eType="#//IPortType"/>
    <eOperations name="createService" eType="#//IService"/>
    <eOperations name="getBinding" eType="#//IBinding">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="getBindings" eType="#//IMap"/>
    <eOperations name="getImports" eType="#//IMap"/>
    <eOperations name="getImports" eType="#//IList">
      <eParameters name="namespaceURI" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="getMessage" eType="#//IMessage">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="getMessages" eType="#//IMap"/>
    <eOperations name="getNamespace" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString">
      <eParameters name="prefix" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="getNamespaces" eType="#//IMap"/>
    <eOperations name="getPortType" eType="#//IPortType">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="getPortTypes" eType="#//IMap"/>
    <eOperations name="getPrefix" eType="ecore:EDataType␣http:
          //www.eclipse.org/emf/2002/Ecore#//EString">
      <eParameters name="namespaceURI" eType="ecore:EDataType␣
          http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="getService" eType="#//IService">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="getServices" eType="#//IMap"/>
    <eOperations name="getExtensionRegistry" eType="#//
        IExtensionRegistry"/>
    <eOperations name="setExtensionRegistry">
      <eParameters name="extensionRegistry" eType="#//
          IExtensionRegistry"/>
    </eOperations>
    <eOperations name="getDocumentBaseURI" eType="
        ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore
        #//EString"/>
    <eOperations name="setDocumentBaseURI">
      <eParameters name="documentBase" eType="ecore:EDataType␣
          http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eOperations>
    <eOperations name="createTypes" eType="#//ITypes"/>
    <eOperations name="removeService" eType="#//IService">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="removeBinding" eType="#//IBinding">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="removePortType" eType="#//IPortType">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="removeMessage" eType="#//IMessage">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="getTypes" eType="#//ITypes"/>
    <eOperations name="setTypes">
      <eParameters name="types" eType="#//ITypes"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IImport"
      instanceClassName="javax.wsdl.Import"
      abstract="true" interface="true" eSuperTypes="#//
          IAttributeExtensible">
    <eAnnotations source="http://www.eclipse.org/emf/2002/
        GenModel">
      <details key="documentation" value="This␣class␣represents
```

```
            ␣the␣javax.wsdl.Import␣non-MOF␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IList"
         instanceClassName="java.util.List"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣non-MOF␣java.util.List␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IMap"
         instanceClassName="java.util.Map"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣non-MOF␣java.util.Map␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IURL"
         instanceClassName="java.net.URL"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣non-MOF␣java.net.URL␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="
         IExtensionRegistry" instanceClassName="javax.wsdl.
         extensions.ExtensionRegistry"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
            ␣the␣non-MOF␣javax.wsdl.extensions.ExtensionRegistry␣
            interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Types"
         eSuperTypes="#//ExtensibleElement␣#//ITypes">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
        <details key="documentation" value="This␣class␣represents
```

```
              ␣a␣WSDL␣types␣element.␣The␣types␣element␣encloses␣
              data␣type␣definitions␣that␣are␣relevant␣for␣the␣
              exchanged␣messages."/>
      </eAnnotations>
      <eOperations name="getSchemas" eType="#//IList"/>
      <eOperations name="getSchemas" eType="#//IList">
        <eParameters name="namespaceURI" eType="ecore:EDataType␣
              http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eOperations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IIterator"
        instanceClassName="java.util.Iterator"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
        <details key="documentation" value="This␣class␣represents
              ␣the␣non-MOF␣java.util.Iterator␣interface."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EDataType" name="WSDLException"
         instanceClassName="javax.wsdl.WSDLException">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
        <details key="documentation" value="This␣class␣represents
              ␣the␣javax.wsdl.Exception␣class."/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="ITypes"
        instanceClassName="javax.wsdl.Types"
        abstract="true" interface="true">
      <eAnnotations source="http://www.eclipse.org/emf/2002/
            GenModel">
        <details key="documentation" value="This␣class␣represents
              ␣the␣javax.wsdl.Types␣non-MOF␣interface"/>
      </eAnnotations>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="
        UnknownExtensibilityElement" eSuperTypes="#//
        ExtensibilityElement">
      <eStructuralFeatures xsi:type="ecore:EReference" name="
            children" upperBound="-1"
          eType="#//UnknownExtensibilityElement" containment="
                true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="
        XSDSchemaExtensibilityElement" eSuperTypes="#//
        ExtensibilityElement␣#//ISchema">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
          documentBaseURI" eType="ecore:EDataType␣http://www.
          eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
          schema" eType="ecore:EClass␣platform:/plugin/org.
          eclipse.xsd/model/XSD.ecore#//XSDSchema"
         containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EDataType" name="DOMDocument"
        instanceClassName="org.w3c.dom.Document">
    <eAnnotations source="http://www.eclipse.org/emf/2002/
          GenModel">
      <details key="documentation" value="This␣class␣represents
            ␣the␣org.w3c.dom.Document␣non-MOF␣interface."/>
    </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="MessageReference"
         abstract="true" eSuperTypes="#//ExtensibleElement">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name
          " eType="ecore:EDataType␣http://www.eclipse.org/emf
          /2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="
          eMessage" lowerBound="1"
         eType="#//Message"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="
        IElementExtensible" instanceClassName="javax.wsdl.
        extensions.ElementExtensible"
       abstract="true" interface="true">
    <eOperations name="getExtensibilityElements" eType="#//
          IList"/>
    <eOperations name="addExtensibilityElement">
      <eParameters name="extElement" eType="#//
            IExtensibilityElement"/>
    </eOperations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="
        IAttributeExtensible" instanceClassName="javax.wsdl.
        extensions.AttributeExtensible"
       abstract="true" interface="true">
    <eOperations name="getExtensionAttribute" eType="#//IObject
          ">
      <eParameters name="name" eType="#//QName"/>
    </eOperations>
    <eOperations name="setExtensionAttribute">
```

```
        <eParameters name="name" eType="#//QName"/>
        <eParameters name="value" eType="#//IObject"/>
      </eOperations>
      <eOperations name="getExtensionAttributes" eType="#//IMap"/
          >
      <eOperations name="getNativeAttributeNames" eType="#//IList
          "/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="IObject"
        instanceClassName="java.lang.Object"
        abstract="true" interface="true"/>
    <eClassifiers xsi:type="ecore:EClass" name="ISchema"
        instanceClassName="javax.wsdl.extensions.schema.Schema"
        abstract="true" interface="true" eSuperTypes="#//
            IExtensibilityElement"/>
</ecore:EPackage>
```

A.2. Transformation Scripts

Listing A.4: Procedural QVT for CSMM to WSDL Transformation

```
import com.siemens.ct.texo.ise.m2m.qvt.oml.javaBlackbox.
    WSDLUtil;
import com.siemens.ct.texo.ise.m2m.qvt.oml.javaBlackbox.XSDUtil
    ;
modeltype CSM uses "http://www.siemens.com/CT/texo/ise/
    ConceptualServiceModel";
modeltype WSDL uses "http://www.eclipse.org/wsdl/2003/WSDL";
modeltype XSD uses "http://www.eclipse.org/xsd/2002/XSD";

transformation CSM2WSDLTransform(in csm: CSM, out wsdl :WSDL);

helper isLogged() : Boolean = false;
helper targetNsPrefix() : String = 'tns';
helper targetNs() : String = 'http://www.example.org/
    NewWSDLFile/';

main() {
    var wsdlDefinition:= csm.objectsOfType(ServiceProduct).map
        serviceProduct2wsdlDefinitions();
}

mapping ServiceProduct::serviceProduct2wsdlDefinitions():wsdl::
    Definition{
```

```
    log ('transformation␣started...');
    result.addNamespace('soap','http://schemas.xmlsoap.org/wsdl
        /soap/');
    result.addNamespace('wsdl','http://schemas.xmlsoap.org/wsdl
        /');
    result.addNamespace('xsd', 'http://www.w3.org/2001/
        XMLSchema');
    result.addNamespace(targetNsPrefix(), targetNs());
    result.targetNamespace :=  targetNs();
    result.qName := getQName(self.name.replace('␣','_'));
    documentationElement := result.createDocumentation('wsdl');
    documentationElement.appendTextContentToDOMElement('[key:'
        + self.key.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        description:' + self.description + ']');
    documentationElement.appendTextContentToDOMElement('[
        documentation:' + self.documentation.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[spkn␣l
        .:' +
        self.spokenLanguage->iterate(l : Lang; langs : String =
            '' |  langs := langs + l.repr() + ';') + ']');
    documentationElement.appendTextContentToDOMElement('[wr␣l.:
        ' +
        self.writtenLanguage->iterate(l : Lang; langs : String
            = '' |  langs := langs + l.repr() + ';') + ']');
    documentationElement.appendTextContentToDOMElement('[
        version:' + self.version.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        created:' + self.created.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        updated:' + self.updated.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[next␣
        update:' +
        self.nextUpdate->iterate(d : String; updates : String =
            '' |  updates := updates + d.repr() + ';') + ']');
    documentationElement.appendTextContentToDOMElement('[type:'
         + self.type.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        automation:' + self.automation.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        composition:' + self.composition.repr() + ']');
    documentationElement.appendTextContentToDOMElement('[
        customizable:' + self.customizable.repr() + ']');

    if csm.objectsOfType(TermsOfUse)->size() > 0 then{
        var tou = csm.objectsOfType(TermsOfUse)->asOrderedSet()->first();
        documentationElement.appendTextContentToDOMElement('[paym.␣cond.:' + tou.paymentCondition.repr() + ']');
        documentationElement.appendTextContentToDOMElement('[deli.␣cond.:' + tou.deliveryCondition.repr() + ']');
    }
    endif;

    csm.objectsOfType(Classification)->forEach(c){
        documentationElement.appendTextContentToDOMElement('[classi.:' + c.value + '␣' + c.system.repr() + ']');
    };
    csm.objectsOfType(Benefit)->forEach(b){
        documentationElement.appendTextContentToDOMElement('[benefit:' + b.name + ']');
    };

    -- begin build xsd schema
    result.setTypes(buildTypeSystem());
    -- end
    result.eMessages += csm.objectsOfType(Capability)
                        ->select(Capability_Condition->exists(metaClassName() = 'PreCondition'))
                        ->map capability2wsdlInputMessage();
    result.eMessages += csm.objectsOfType(Capability)
                        ->select(Capability_Condition->exists(metaClassName() = 'PostCondition'))
                        ->map capability2wsdlOutputMessage();
    result.ePortTypes += self.map buildPortType();
    log ('transformation␣done!');
}

mapping Capability::capability2wsdlInputMessage() : wsdl::Message{
    result.qName := getQName(self.name.replace('␣','_') + 'Input');
    self.Capability_Condition->forEach(c | c.metaClassName()= 'PreCondition'
        and (not c.Condition_ref_Resource.oclIsUndefined())){
        result.eParts += object Part{
            name := c.name.replace('␣','_');
            elementName := getQName(targetNsPrefix() + ':' +
                               c.Condition_ref_Resource.name.replace('␣','_'));
        }
    };
}

mapping Capability::capability2wsdlOutputMessage() : wsdl::Message{
    result.qName := getQName(self.name.replace('␣','_') + 'Output');
    self.Capability_Condition->forEach(c | c.metaClassName()= 'PostCondition'
        and (not c.Condition_ref_Resource.oclIsUndefined())){
        result.eParts += object Part{
            name := c.name.replace('␣','_');
            elementName := getQName(targetNsPrefix() + ':' +
                            c.Condition_ref_Resource.name.replace('␣','_'))
        }
    };
}

mapping ServiceProduct::buildPortType() : wsdl::PortType{
    qName := getQName(self.name.replace('␣','_'));
    self.SProduct_Capability->forEach(c){
         eOperations += object Operation{
             name := c.name.replace('␣','_');
             var inputMsgName := name + 'Input';
             var outputMsgName := name + 'Output';
             -- create documentation for operation
             documentationElement := createDOMElement('wsdl', 'documentation');
             documentationElement.appendTextContentToDOMElement(c.description);
            -- end
             if (wsdl.objectsOfType(Message)->exists(qName.repr() = inputMsgName)) then
                 {    eInput := object Input{
                     --eMessage := wsdl.objectsOfType(Message)->any(qName.repr() = inputMsgName);
                     };
                     eInput.setMessageRef(targetNsPrefix() + ':' + inputMsgName);
                 }
             endif;
             if (wsdl.objectsOfType(Message)->exists(qName.repr() = outputMsgName)) then
                 {    eOutput := object Output{
                    --eMessage := wsdl.objectsOfType(Message)->any(qName.repr() = outputMsgName);
                    };
                    eOutput.setMessageRef(targetNsPrefix() + ':' + outputMsgName);
                }
            endif;
        }
    }
}

query buildTypeSystem() : Types{
    var types := object Types{};
    var extElement := object XSDSchemaExtensibilityElement{};
    var xsdSchema := object XSDSchema{ targetNamespace := targetNs()};
    -- call xsdUtil to set prefix of schema
    xsdSchema.setSchemaForSchemaQNamePrefix('xsd');
    xsdSchema.putIntoQNamePrefixToNamespaceMap('xsd', 'http://www.w3.org/2001/XMLSchema');
    -- call end
    xsdSchema.contents += csm.objectsOfType(Resource)->map resource2xsdElement(xsdSchema);

    extElement.schema := xsdSchema;
    types.addExtensibilityElement(extElement);
    return types;
}

mapping Resource::resource2xsdElement(xsdSchema : XSDSchema): xsd::XSDElementDeclaration{
    name := self.name.replace(' ','_');
    annotation := object XSDAnnotation{};
    annotation.createUserInformation(xsdSchema, result, 'documentation', self.description);
    annotation.createUserInformation(xsdSchema, result, 'resourcetype', self.type.repr());
    anonymousTypeDefinition := object xsd::XSDComplexTypeDefinition{
        var modelGroup := object xsd::XSDModelGroup{
            compositor := xsd::XSDCompositor::sequence;
        };
        var particle := object xsd::XSDParticle{
            content := modelGroup;
        };
        content := particle;
    };
}
```

Listing A.5: Declarative QVT for CSMM to WSDL Transformation

```

transformation CSM2WSDLTransform
        (csm : ConceptualServiceModel, wsdl : wsdl, m2mutil : m2mutil){

        query targetNsPrefix():String{'tns'}
        query targetNs():String{'http://www.example.org/NewWSDLFile/'}

        top relation serviceProduct2wsdlDefinition {
                sPKey : String;
                sPDocu : String;
                sPName : String;
                sPDesc : String;
                sPSpkLang : String;
                sPWrtLang : String;
                sPVersion : String;
                sPCreated : String;
                sPUpdated : String;
                sPNextUpd : String;
                sPType : ConceptualServiceModel::ServiceType;
                sPAutomat : ConceptualServiceModel::AutomationLevel;
                sPCompos : ConceptualServiceModel::CompositionLevel;
                sPCustom : Boolean;
                checkonly domain csm  serviceProduct: ConceptualServiceModel::ServiceProduct{
                    name = sPName,
                    description = sPDesc,
                    type = sPType,
                    automation = sPAutomat,
                    composition = sPCompos,
                    customizable = sPCustom
                };
                enforce domain wsdl wsdlDefinition:wsdl::Definition{
                    targetNamespace = targetNs(),
                    eTypes =  types : wsdl::Types {
                            eExtensibilityElements = wsdlExtElement : wsdl::
                                    XSDSchemaExtensibilityElement
                                     {
                                        schema = xsdSchema :
                                            xsd::XSDSchema {
                                                targetNamespace
                                                    = targetNs
                                                    ()
                                        }
                                }
                        },
                        ePortTypes = portType : wsdl::PortType
                            {}
                };
                when{
                        /* the 'getAttributeValue' method is a
                            workaround solution using EMF's
                            reflection mechanism
                        * to get the value of attribute, for
                            the case that the transformation
                            engine doesn't recognize
                        * EDataType defined in ecore model,
                            this could be a bug of medini QVT
                            since it is migrated onto
                        * Galileo.
                        */
                        sPKey = getSPAttributeValue(
                            serviceProduct, 'key');
                        sPDocu = getSPAttributeValue(
                            serviceProduct, 'documentation');
                        sPSpkLang = getSPAttributeValue(
                            serviceProduct, 'spokenLanguage');
                        sPWrtLang = getSPAttributeValue(
                            serviceProduct, 'writtenLanguage');
                        sPVersion = getSPAttributeValue(
                            serviceProduct, 'version');
                        sPCreated = getSPAttributeValue(
                            serviceProduct, 'created');
                        sPUpdated = getSPAttributeValue(
                            serviceProduct, 'updated');
                        sPNextUpd = getSPAttributeValue(
                            serviceProduct, 'nextUpdate');
                }
                where{
                        createWSDLDocument(wsdlDefinition);
                        addNamespace(wsdlDefinition, 'soap', '
                            http://schemas.xmlsoap.org/wsdl/
```

```
          soap/');
        addNamespace(wsdlDefinition,'wsdl','
          http://schemas.xmlsoap.org/wsdl/');
        addNamespace(wsdlDefinition,'xsd','http
          ://www.w3.org/2001/XMLSchema');
        addNamespace(wsdlDefinition,
          targetNsPrefix(),targetNs());
        setQName(wsdlDefinition, sPName.replace
          (' ','_'));
        createDocumentation(wsdlDefinition, '
          wsdl');
        appendDocuText(wsdlDefinition, '[key:'.
          concat(sPKey).concat(']'));
        appendDocuText(wsdlDefinition, '[
          description:'.concat(sPDesc).concat
          (']'));
        appendDocuText(wsdlDefinition, '[
          documentation:'.concat(sPDocu).
          concat(']'));
        appendDocuText(wsdlDefinition, '[spkn l
          .:'.concat(sPSpkLang).concat(']'));
        appendDocuText(wsdlDefinition, '[wr l.:
          '.concat(sPWrtLang).concat(']'));
        appendDocuText(wsdlDefinition, '[
          version:'.concat(sPVersion).concat(
          ']'));
        appendDocuText(wsdlDefinition, '[
          created:'.concat(sPCreated).concat(
          ']'));
        appendDocuText(wsdlDefinition, '[
          updated:'.concat(sPUpdated).concat(
          ']'));
        appendDocuText(wsdlDefinition, '[next 
          update:'.concat(sPNextUpd).concat('
          ]'));
        appendDocuText(wsdlDefinition, '[type:'
          .concat(getServiceTypeValue(sPType)
          ).concat(']'));
        appendDocuText(wsdlDefinition, '[
          automation:'.concat(
          getAutomationLevelValue(sPAutomat))
          .concat(']'));
        appendDocuText(wsdlDefinition, '[
          composition:'.concat(
          getCompositionLevelValue(sPCompos))
          .concat(']'));
```

```
            appendDocuText(wsdlDefinition, '[customizable:'.concat(if sPCustom then 'True' else 'False' endif).concat(']'));

            if TermsOfUse.allInstances()->size()>0 then
                appendDocuText(wsdlDefinition, '[paym.␣cond.:'.concat(getTOUAttributeValue
                    (TermsOfUse.allInstances()->asSequence()->first(),'paymentCondition')).concat(']'))
                and
                appendDocuText(wsdlDefinition, '[deli.␣cond.:'.concat(getTOUAttributeValue
                    (TermsOfUse.allInstances()->asSequence()->first(),'deliveryCondition')).concat(']'))
            else true
            endif;

            if Classification.allInstances()->size()>0 then
                appendDocuText(wsdlDefinition, getClassificationValue())
            else
                true
            endif;

            if Benefit.allInstances()->size()>0 then
                appendDocuText(wsdlDefinition, getBenefitValue())
            else true
            endif;

            setQName(portType, sPName.replace('␣','_'));
```

```
                resource2xsdElement(serviceProduct, xsdSchema);

                capability2inputMessage(serviceProduct, wsdlDefinition);
                capability2outputMessage(serviceProduct, wsdlDefinition);
                capability2wsdlOperation(serviceProduct, portType);
            }
    }

    relation resource2xsdElement {
        resourceName : String;
        resourceDesc : String;
        resourceType : ConceptualServiceModel::ResourceType;
        checkonly domain csm serviceProduct: ConceptualServiceModel::ServiceProduct  {
            SProduct_Resource = resource : ConceptualServiceModel::Resource {
                name = resourceName,
                description = resourceDesc,
                type = resourceType
            }
        };
        enforce domain wsdl xsdSchema : xsd::XSDSchema {
            contents = xsdElement : xsd::XSDElementDeclaration {
                name = resourceName.replace('␣','_'),
                annotation = anno : xsd::XSDAnnotation {},
                anonymousTypeDefinition = schemaType : xsd::XSDComplexTypeDefinition {
                    --content = xsdParticle : xsd::XSDParticle {
                    --      content = modelGroup : xsd::XSDModelGroup {
                    --  compositor = xsd::XSDCompositor::
                        sequence
                    --      }
                    --}
                }
            }
        };
        where{
            setSchemaForSchemaQNamePrefix(xsdSchema
              , 'xsd');
            putIntoQNamePrefixToNamespaceMap(
              xsdSchema,'xsd','http://www.w3.org
              /2001/XMLSchema');

            createUserInformation(anno, xsdSchema,
              xsdElement, 'documentation',
              resourceDesc);
            createUserInformation(anno, xsdSchema,
              xsdElement, 'resourcetype',
              getResourceTypeValue(resourceType))
              ;
        }
    }

    relation capability2inputMessage {
        capaName : String;
        checkonly domain csm serviceProduct:
          ConceptualServiceModel::ServiceProduct  {
            SProduct_Capability = capability :
              ConceptualServiceModel::Capability
              {
                name = capaName
            }
        };
        enforce domain wsdl wsdlDefinition:wsdl::
          Definition  {
            eMessages = msg : wsdl::Message {}
        };
        when{
            capability.Capability_Condition->exists
              (con : Condition | con.oclIsTypeOf(
              PreCondition));
        }
        where{
            setQName(msg, capaName.replace('␣','_')
               + 'Input');
            preCondition2part(capability, msg);
        }
    }

    relation preCondition2part{
        condName : String;
        resoName : String;
        checkonly domain csm capability : ConceptualServiceModel::Capability {
            Capability_Condition = condition: ConceptualServiceModel::PreCondition{
                name = condName,
                Condition_ref_Resource = resource : ConceptualServiceModel::Resource {
                    name = resoName
                }
            }
        };
        enforce domain wsdl msg:wsdl::Message{
            eParts = part : wsdl::Part {
                name = condName.replace(’␣’,’_’)
            }
        };
        where{
            setQName(part, targetNsPrefix().concat(’:’).concat(resoName.replace(’␣’,’_’)));
        }
    }

    relation capability2outputMessage {
        capaName : String;
        checkonly domain csm serviceProduct: ConceptualServiceModel::ServiceProduct  {
            SProduct_Capability = capability : ConceptualServiceModel::Capability {
                name = capaName
            }
        };
        enforce domain wsdl wsdlDefinition:wsdl::Definition  {
            eMessages = msg : wsdl::Message {}
        };
        when{
            capability.Capability_Condition->exists(con : Condition | con.oclIsTypeOf(PostCondition));
        }
        where{
            setQName(msg, capaName.replace('␣','_') + 'Output');
            postCondition2part(capability, msg);
        }
    }

    relation postCondition2part{
        condName : String;
        resoName : String;
        checkonly domain csm capability : ConceptualServiceModel::Capability {
            Capability_Condition = condition:ConceptualServiceModel::PostCondition{
                name = condName,
                Condition_ref_Resource = resource : ConceptualServiceModel::Resource {
                    name = resoName
                }
            }
        };
        enforce domain wsdl msg:wsdl::Message{
            eParts = part : wsdl::Part {
                name = condName.replace('␣','_')
            }
        };
        where{
            setQName(part, targetNsPrefix().concat(':').concat(resoName.replace('␣','_')));
        }
    }

    relation capability2wsdlOperation {
        capaName : String;
        capaDesc : String;
```

```
            opName : String;
            enforce domain csm serviceProduct: ConceptualServiceModel::ServiceProduct {
                SProduct_Capability = capability : ConceptualServiceModel::Capability {
                    name = capaName,
                    description = capaDesc
                }
            };
            enforce domain wsdl wsdlPortType:wsdl::PortType {
                eOperations = operation : wsdl::Operation {
                    name = opName,
                    eInput = input : wsdl::Input {},
                    eOutput = output : wsdl::Output {}
                }
            };
            where{
                opName = capaName.replace('␣','_');
                createDocumentation(operation, 'wsdl');
                appendDocuText(operation,capaDesc);
                setInputMessage(input,targetNsPrefix().concat(':').concat(opName).concat('Input'));
                setOutputMessage(output,targetNsPrefix().concat(':').concat(opName).concat('Output'));
            }
        }

        query getClassificationValue() : String {
            Classification.allInstances()->iterate(clsf : Classification; default : String = '' |
                default.concat('[classi.:').concat(clsf.value).concat('␣').concat
                (getClassificationSystemValue(clsf.system)).concat(']'))
        }

        query getBenefitValue() : String{
            Benefit.allInstances()->iterate(bfit : Benefit; default : String = '' |
                        default.concat('[benefit:').concat(bfit
                            .name).concat(']'))
        }

        -- hidden feature to printing log info on screen
        query _debug(log : String) : String {
                log
        }

        /**
        * call custom code written in java
        **/
        -- call setQName
        query setQName (element : WSDLElement, value : String)
            : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().setQName(element, value)
        }
        -- call createWSDLDocument
        query createWSDLDocument (wsdlDefinition : wsdl::
            Definition) : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().createWSDLDocument(wsdlDefinition)
        }
        -- call addNamespace
        query addNamespace (wsdlDefinition : wsdl::Definition,
            prefix : String, ns : String) : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().addNamespace(wsdlDefinition,prefix,
                    ns)
        }
        -- call createDocumentation
        query createDocumentation (wsdlElement : wsdl::
            WSDLElement, prefix : String) : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().createDocumentation(wsdlElement,
                    prefix)
        }
        -- call appendDocuText
        query appendDocuText (wsdlElement : wsdl::WSDLElement,
            textContent : String) : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().appendDocuText(wsdlElement,
                    textContent)
        }
        -- call getEAttributeByName
        query getSPAttributeValue (sp : ConceptualServiceModel
             ::ServiceProduct, attrName : String) : String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEAttributeByName(sp,attrName)
        }
        query getTOUAttributeValue (tou :
             ConceptualServiceModel::TermsOfUse, attrName :
             String) : String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEAttributeByName(tou,attrName)
        }
        -- call getEnumLiteral
        query getServiceTypeValue (st : ServiceType) : String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEnumLiteral(st)
        }
        query getAutomationLevelValue (al : AutomationLevel) :
             String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEnumLiteral(al)
        }
        query getCompositionLevelValue (cl : CompositionLevel)
             : String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEnumLiteral(cl)
        }
        query getClassificationSystemValue (cs :
             ClassificationSystem) : String {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEnumLiteral(cs)
        }
        query getResourceTypeValue (rt : ResourceType) : String
              {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().getEnumLiteral(rt)
        }
        -- call setMessageRef
        query setInputMessage(wsdlInput : wsdl::Input, msgName
             : String) : Boolean{
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().setMessageRef(wsdlInput,msgName)
        }
        query setOutputMessage(wsdlOutput : wsdl::Output,
             msgName : String) : Boolean{
                m2mutil::M2MUtil.allInstances()->asSequence()->
                     first().setMessageRef(wsdlOutput,msgName)
        }
        -- call methods defined for xsd
        query setSchemaForSchemaQNamePrefix (schema : xsd::
            XSDSchema, prefix : String) : Boolean {
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().setSchemaForSchemaQNamePrefix(
                    schema,prefix)
        }
        query putIntoQNamePrefixToNamespaceMap(schema : xsd::
            XSDSchema, prefix : String, namespace : String) :
            Boolean{
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().putIntoQNamePrefixToNamespaceMap(
                    schema,prefix,namespace)
        }
        -- call createUserInformation
        query createUserInformation(annotation : xsd::
            XSDAnnotation, schema : xsd::XSDSchema, xsdElement
            : xsd::XSDElementDeclaration,
                        sourceURI : String, docu : String) :
                            Boolean{
                m2mutil::M2MUtil.allInstances()->asSequence()->
                    first().createUserInformation(annotation,
                    schema, xsdElement, sourceURI, docu)
        }
}
```

Listing A.6: ATL for CSMM to WSDL Transformation

```
-- @nsURI M2MUtil=http://www.siemens.com/CT/texo/ise/m2mutil
-- @nsURI WSDL=http://www.eclipse.org/wsdl/2003/WSDL
-- @nsURI CSM=http://www.siemens.com/CT/texo/ise/
    ConceptualServiceModel
-- @nsURI XSD=http://www.eclipse.org/xsd/2002/XSD

-- !!!IMPORTANT!!!
-- In order to run this transformation, it is provided that a
    bug must be fixed in the
-- current version of plugin 'org.eclipse.wst.wsdl' (1.2.2.
    v200911111930), which can be
-- achieved by copying the modified version under /lib into the
     eclipse's␣plugins␣ordner
--␣to␣overwrite␣the␣original.

module␣CSM2WSDLTransform;
create␣OUT␣:␣WSDL␣from␣IN1␣:␣CSM,␣IN2␣:␣M2MUtil;
```

```

helper␣def␣:␣isLogged␣:␣Boolean␣=␣true;
helper␣def␣:␣targetNsPrefix␣:␣String␣=␣'tns';
helper␣def␣:␣targetNs␣:␣String␣=␣'http://www.example.org/NewWSDLFile/';

helper def : m2mutil : M2MUtil!M2MUtil = M2MUtil!M2MUtil.allInstances()->asSequence()->first();

rule serviceProduct2wsdlDefinition{

        from
                serviceProduct : CSM!ServiceProduct
        to
                xsdSchema : WSDL!XSDSchema (
                        targetNamespace <- thisModule.targetNs
                ),
                extSchema : WSDL!XSDSchemaExtensibilityElement (
                        schema <- xsdSchema
                ),
                typeSys : WSDL!Types(
                        eExtensibilityElements <- extSchema
                ),
                portType : WSDL!PortType(
                        eOperations <- CSM!Capability.allInstances()
                ),
                wsdlDefinition : WSDL!Definition(
                        targetNamespace <- thisModule.targetNs.replace('␣','_'),
                        eTypes <- typeSys,
                        ePortTypes <- portType
                )
        do{
                thisModule.m2mutil.createDocumentation(wsdlDefinition, 'wsdl');
                thisModule.m2mutil.setQName(wsdlDefinition, serviceProduct.name.replace('␣','_'));
                thisModule.m2mutil.setSchemaForSchemaQNamePrefix(extSchema.schema, 'xsd');
                thisModule.m2mutil.putIntoQNamePrefixToNamespaceMap(extSchema.schema,'xsd','http://www.w3.org/2001/XMLSchema');
```

```
            thisModule.m2mutil.setQName(portType, serviceProduct.name.replace('␣','_'));
            wsdlDefinition.addNamespace('soap','http://schemas.xmlsoap.org/wsdl/soap/');
            wsdlDefinition.addNamespace('wsdl','http://schemas.xmlsoap.org/wsdl/');
            wsdlDefinition.addNamespace('xsd', 'http://www.w3.org/2001/XMLSchema');
            wsdlDefinition.addNamespace(thisModule.targetNsPrefix, thisModule.targetNs);
            if (thisModule.isLogged) '->␣WSDL␣Definition␣is␣created'.println();
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[key:'.concat(serviceProduct.key).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[description:'.concat(serviceProduct.description).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[documentation:'.concat(serviceProduct.documentation).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[spkn␣l.:'.concat(serviceProduct.spokenLanguage).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[wr␣l.:'.concat(serviceProduct.writtenLanguage).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[version:'.concat(serviceProduct.version).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[created:'.concat(serviceProduct.created).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[updated:'.concat(serviceProduct.updated).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[next␣update:'.concat(serviceProduct.nextUpdate).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[type:'.concat(serviceProduct.type).concat(']'));
            thisModule.m2mutil.appendDocuText(wsdlDefinition, '[automation:'.concat(
            serviceProduct.automation).concat(']'));
        thisModule.m2mutil.appendDocuText(
            wsdlDefinition, '[composition:'.concat(
            serviceProduct.composition).concat(']'));
        thisModule.m2mutil.appendDocuText(
            wsdlDefinition, '[customizable:'.concat(
            serviceProduct.customizable).concat(']'));

        if (CSM!TermsOfUse.allInstances()->size() > 0){
                thisModule.m2mutil.appendDocuText(
                    wsdlDefinition,
                        '[paym.␣cond.:' + CSM!
                            TermsOfUse.allInstances()->
                            asOrderedSet()->first().
                            paymentCondition + ']');
                thisModule.m2mutil.appendDocuText(
                    wsdlDefinition,
                        '[deli.␣cond.:' + CSM!
                            TermsOfUse.allInstances()->
                            asOrderedSet()->first().
                            deliveryCondition + ']');
        }

        for (cl in CSM!Classification.allInstances()){
                thisModule.m2mutil.appendDocuText(
                    wsdlDefinition,'[classi.:' + cl.
                    value + '␣' + cl.system + ']');
        }
        for (be in CSM!Benefit.allInstances()){
                thisModule.m2mutil.appendDocuText(
                    wsdlDefinition,'[benefit:' + be.
                    name + ']');
        }
        -- call lazy rule 'resource2xsdElement'
        -- xsdSchema.contents <- CSM!Resource.
            allInstances()->collect(r | thisModule.
            resource2xsdElement(r));
        -- call called rule 'resource2xsdElement'
        for(res in CSM!Resource.allInstances()){
                thisModule.resource2xsdElement(res,
                    xsdSchema);
        }

        -- call lazy rule 'capability2wsdlInputMessage'
        wsdlDefinition.eMessages <- CSM!Capability.
            allInstances()->select(c | c.
                Capability_Condition
                                ->exists(con | con.oclIsTypeOf(CSM!PreCondition)))
                                ->collect(capa | thisModule.capability2wsdlInputMessage(capa));
            -- call lazy rule 'capability2wsdlOutputMessage'
            wsdlDefinition.eMessages <- CSM!Capability.allInstances()->select(c | c.Capability_Condition
                                ->exists(con | con.oclIsTypeOf(CSM!PostCondition)))
                                ->collect(capa | thisModule.capability2wsdlOutputMessage(capa));
        }
}

lazy rule capability2wsdlInputMessage{
        from capa : CSM!Capability
        to
            inputMsg : WSDL!Message(
                    eParts <- capa.Capability_Condition->select(preCond | preCond.oclIsTypeOf(CSM!PreCondition) and
                            (not preCond.Condition_ref_Resource.oclIsUndefined()))
            )
        do{
            thisModule.m2mutil.setQName(inputMsg, capa.name.replace('␣','_').concat('Input'));
            if (thisModule.isLogged) '->␣Input␣message␣"'.concat(inputMsg.qName).concat('"␣is␣created').println();
        }
}

lazy rule capability2wsdlOutputMessage{
        from capa : CSM!Capability
        to
            outputMsg : WSDL!Message(
                    eParts <- capa.Capability_Condition->select(postCond | postCond.oclIsTypeOf(CSM!PostCondition) and
                (not postCond.Condition_ref_Resource.oclIsUndefined()))
        )
    do{
        thisModule.m2mutil.setQName(outputMsg, capa.name.replace('␣','_').concat('Output'));
        if (thisModule.isLogged) '->␣Output␣message␣"'.concat(outputMsg.qName).concat('"␣is␣created').println();
    }
}

rule condition2part{
    from cond : CSM!Condition
    to
        part : WSDL!Part(
            name <- cond.name.replace('␣','_')
        )
    do{
        thisModule.m2mutil.setQName(part,thisModule.targetNsPrefix.concat(':')
            .concat(cond.Condition_ref_Resource.name.replace('␣','_')));
        if (thisModule.isLogged) '->␣Part␣"'.concat(part.name).concat('"␣is␣created').println();
    }
}

rule resource2xsdElement (resource : CSM!Resource, schema : WSDL!XSDSchema){
    using{
        res : CSM!Resource = resource;
    }
    to
        xsdElement : WSDL!XSDElementDeclaration(
            name <- resource.name.replace('␣','_'),
            annotation <- xsdAnnotation,
            anonymousTypeDefinition <- complexType
        ),
        complexType : WSDL!XSDComplexTypeDefinition(
            content <- particle
        ),
        particle : WSDL!XSDParticle(
            content <- modelGroup
            ),
            modelGroup : WSDL!XSDModelGroup(
                compositor <- #sequence
            ),
            xsdAnnotation : WSDL!XSDAnnotation
        do{
            if (thisModule.isLogged) '->␣Schema␣element␣"'.concat(xsdElement.name).concat('"␣is␣created').println();
            thisModule.m2mutil.createUserInformation(xsdAnnotation, schema, xsdElement, 'documentation',
                if resource.description.oclIsUndefined() then 'null' else resource.description endif);
            thisModule.m2mutil.createUserInformation(xsdAnnotation, schema, xsdElement, 'resourcetype',
                resource.type.toString());
        }
}

rule capability2wsdlOperation{
        from capa : CSM!Capability
        to
            operation : WSDL!Operation(
                name <- capa.name.replace('␣', '_'),
                eInput <- input,
                eOutput<- output
            ),
            input : WSDL!Input,
            output : WSDL!Output
        do{
            thisModule.m2mutil.createDocumentation(operation, 'wsdl');
            thisModule.m2mutil.appendDocuText(operation, if not capa.description.oclIsUndefined()
                then capa.description else 'null' endif);
            thisModule.m2mutil.setMessageRef(input, thisModule.targetNsPrefix.concat(':')
                .concat(operation.name).concat('Input'));
            thisModule.m2mutil.setMessageRef(output, thisModule.targetNsPrefix.concat(':')
                .concat(operation.name).concat('Output'
                                ));
                        if (thisModule.isLogged) '-> Operation "'.
                            concat(operation.name).concat('" is created
                            ').println();
            }
    }
```

A.3. Other Listings

Listing A.7: WSDL Structure (modified version from [27])

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>

    <import namespace="uri" location="uri"/>*

    <wsdl:documentation .... /> ?

    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xsd:schema .... />*
    </wsdl:types>

    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>

    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken" message="qname"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input> ?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output> ?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="nmtoken"> *
        <wsdl:documentation .... />?
        <wsdl:port name="nmtoken" binding="qname"> *
           <wsdl:documentation .... /> ?
        </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

A.4. Case Study and Example Result Documents (XMI)

Eco Value Calculation Example

Listing A.8: Business Service Diagram XML for Eco Value Calculation Example

```
<?xml version="1.0" encoding="UTF-8"?>
<bsm:ValueModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:bsm="http://www.siemens.com/CT/texo/ise/BusinessServiceModel" author="Gregor␣Scheithauer" created="2010-04-15" version="1">
  <RevenueModel revenueModelID="_YuqKQEiREd-CfbCAB98smw" name="for␣Producing␣Companies" streamType="Selling" pricingMethod="Fixed␣Price" RM_TC="_T7sxkEiREd-CfbCAB98smw"/>
  <TargetCustomer targetCustomerID="_T7sxkEiREd-CfbCAB98smw" name="Producing␣Companies">
```

```
    <consistsOf customerEquity="Acquisition"/>
  </TargetCustomer>
  <ValueObject valueObjectID="_HfhkwEiREd-CfbCAB98smw" name="
      Certificate" type="Tangible"/>
  <ValueObject valueObjectID="_KyrXkEiREd-CfbCAB98smw" name="
      Individual␣Eco␣Value" type="Intangible"/>
  <DistributionChannel distributionChannelID="_cZgEUEiREd-
      CfbCAB98smw" name="TEXO␣Service␣Marketplace"
      customerBuyingCycle="Purchase"/>
  <ValueOffer name="Eco␣Value␣Calculation" reasoning="Service␣
      Usage" valueLevel="Commodity" priceLevel="Economic"
      lifeCycleStep="Value␣Use" VOf_DC="_cZgEUEiREd-CfbCAB98smw
      " VOf_RM="_YuqKQEiREd-CfbCAB98smw" VOf_TC="_T7sxkEiREd-
      CfbCAB98smw" VOf_VOb="_HfhkwEiREd-CfbCAB98smw␣_KyrXkEiREd
      -CfbCAB98smw"/>
</bsm:ValueModel>
```

Listing A.9: Conceputal Service Diagram XML for Eco Value Calculation Example

```
<?xml version="1.0" encoding="UTF-8"?>
<csm:CSM xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:csm="http://www.siemens.com/CT/texo/ise/
    ConceptualServiceModel">
  <CSM_SProduct name="Eco␣Value␣Calculation" key="
      SVC_KEY_ECOCALC_7493" description="calculates␣the␣carbon␣
      dioxide␣footprint␣for␣any␣given␣material" version="2"
      created="2010-03-09T00:00:00.000+0100" updated="
      2010-03-09T00:00:00.000+0100" composition="Final␣Service"
      >
    <spokenLanguage>eng</spokenLanguage>
    <writtenLanguage>eng</writtenLanguage>
    <nextUpdate>2010-11-10T00:00:00.000+0100</nextUpdate>
    <SProduct_Payment name="Bank␣Transfer" instrument="Bank␣
        Transfer" preferred="true" recurrence="Monthly"/>
    <SProduct_Standard provider="ISO" title="9001" status="N/A"
         created="2008-11-15T00:00:00.000+0100" author="N/A"
        version="2008"/>
    <SProduct_Classification value="12142104" system="UNSPSC"/>
    <SProduct_TermsOfUse paymentCondition="http://62.52.175.245
        :8080/texo/services/ecocalc/paymentcondition"
        deliveryCondition="http://62.52.175.245:8080/texo/
        services/ecocalc/deliverycondition"/>
    <SProduct_Actor xsi:type="csm:Provider" name="CDE␣GmbH" key
```

```
        ="ACT_KEY_3" DUNS="N/A" industry="Carbon␣Dioxide␣
        Manufactoring">
      <Actor_Contact personName="C.␣Schubert" phone="N/A" email
          ="N/A">
        <Contact_ALine keyName="Street" keyValue="Hertz␣Str.␣19
            "/>
        <Contact_ALine keyName="ZIP" keyValue="75015"/>
        <Contact_ALine keyName="City" keyValue="Bretten"/>
        <Contact_ALine keyName="Country" keyValue="Germany"/>
      </Actor_Contact>
    </SProduct_Actor>
    <SProduct_Actor xsi:type="csm:Partner" name="EDS" key="
        ACT_KEY_1" DUNS="N/A" industry="Other␣Information␣
        Services">
      <Actor_Contact personName="Noname1" phone="N/A" email="N/
          A">
        <Contact_ALine keyName="Street" keyValue="Mailbox␣1234"
            />
        <Contact_ALine keyName="ZIP" keyValue="65402"/>
        <Contact_ALine keyName="City" keyValue="Ruesselsheim"/>
        <Contact_ALine keyName="Country" keyValue="Germany"/>
      </Actor_Contact>
    </SProduct_Actor>
    <SProduct_Actor xsi:type="csm:Partner" name="Indian␣
        Chemistry" key="ACT_KEY_2" DUNS="N/A" industry="Natural
        ␣Gas␣Distribution">
      <Actor_Contact personName="Radhike␣Srinagar" phone="N/A"
          email="N/A">
        <Contact_ALine keyName="Street" keyValue="Mah.␣G.␣Ave"/
            >
        <Contact_ALine keyName="ZIP" keyValue="400093"/>
        <Contact_ALine keyName="City" keyValue="Mumbai"/>
        <Contact_ALine keyName="Country" keyValue="India"/>
      </Actor_Contact>
    </SProduct_Actor>
    <SProduct_Resource name="Material"/>
    <SProduct_Resource name="Individual␣Eco␣Value" type="
        Information"/>
    <SProduct_Resource name="Certificate"/>
    <SProduct_Quality name="Calculation␣Quality">
      <Quality_Performance capacity="20">
        <Performance_Latency value="5" granularity="Second"/>
        <Performance_Throughput events="600" recurrence="Every␣
            Second"/>
      </Quality_Performance>
      <Quality_Dependability availability="0.985" reliability="
```

```
          65" reliabilityGran="Hour" maintainability="1"
          maintainaGran="Hour" accuracy="1"/>
      <Quality_Security authorization="true">
        <Security_DIntegrity value="600"/>
      </Quality_Security>
    </SProduct_Quality>
    <SProduct_Price xsi:type="csm:Flatrate" name="flat␣exact"
        amount="300.0" tax="0.19" recurrence="Monthly"/>
    <SProduct_Price xsi:type="csm:TwoPartTariff" name="Monthly␣
        Two␣Part" amount="14.0" tax="0.19" fixedSum="100.0"/>
    <SProduct_Price xsi:type="csm:UsageBased" name="usage-based
        ␣exact" amount="15.0" tax="0.19"/>
    <SProduct_Price xsi:type="csm:NBlockTariff" name="500␣Block
        " amount="16.0" tax="0.19" n="500" allowancePoints="0.1
        "/>
    <SProduct_Channel name="TEXO␣Service␣Market␣Place"
        channelLength="1" productVariety="150" waitingTime="160
        " waitingTimeGran="Second" type="Electronically"/>
    <SProduct_Capability name="Calculate␣Carbon␣Dioxide"
        interface="Technical␣Interface" duration="1"
        durationGran="Minute" Capability_ref_Price="//
        @CSM_SProduct.0/@SProduct_Price.0"
        Capability_ref_Quality="//@CSM_SProduct.0/
        @SProduct_Quality.0">
      <Capability_Condition xsi:type="csm:PreCondition" name="
          Available␣Material" state="available"
          Condition_ref_Resource="//@CSM_SProduct.0/
          @SProduct_Resource.0"/>
      <Capability_Condition xsi:type="csm:PostCondition" name="
          Calculated␣Eco␣Value" state="calculated"
          Condition_ref_Resource="//@CSM_SProduct.0/
          @SProduct_Resource.1"/>
    </SProduct_Capability>
    <SProduct_Capability name="Issue␣Carbon␣Dioxide␣Certificate
        " interface="Technical␣Interface" duration="1"
        durationGran="Hour" Capability_ref_Price="//
        @CSM_SProduct.0/@SProduct_Price.3␣//@CSM_SProduct.0/
        @SProduct_Price.1␣//@CSM_SProduct.0/@SProduct_Price.2">
      <Capability_Condition xsi:type="csm:PreCondition" name="
          Calculated␣Eco␣Value" state="calculated"
          Condition_ref_Resource="//@CSM_SProduct.0/
          @SProduct_Resource.1"/>
      <Capability_Condition xsi:type="csm:PostCondition" name="
          Issued␣Certificate" state="issued"
          Condition_ref_Resource="//@CSM_SProduct.0/
          @SProduct_Resource.2"/>
    </SProduct_Capability>
  </CSM_SProduct>
</csm:CSM>
```

Listing A.10: WSDL for Eco Value Calculation Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:tns="http://www.example.org/NewWSDLFile/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema" name="
    Eco_Value_Calculation" targetNamespace="http://www.example.
    org/NewWSDLFile/">
  <wsdl:documentation>[key:SVC_KEY_ECOCALC_7493][
      description:calculates the carbon dioxide footprint for
      any given material][documentation:org.eclipse.emf.ecore.
      impl.DynamicEObjectImpl@160a70b (eClass: org.eclipse.emf.
      ecore.impl.EClassImpl@76dc80 (name: Invalid_Class) (
      instanceClassName: null) (abstract: false, interface:
      false))][spkn l.:eng;][wr l.:eng;][version:2][created:Tue
       Mar 09 00:00:00 CET 2010][updated:Tue Mar 09 00:00:00
      CET 2010][next update:Wed Nov 10 00:00:00 CET 2010;][
      type:Core Service][automation:Fully Automated][
      composition:Final Service][customizable:false][paym. cond
      .:http://62.52.175.245:8080/texo/services/ecocalc/
      paymentcondition][deli. cond.:http://62.52.175.245:8080/
      texo/services/ecocalc/deliverycondition][classi.:12142104
       UNSPSC]</wsdl:documentation>
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.example.org/
        NewWSDLFile/">
      <xsd:element name="Individual_Eco_Value">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Certificate">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Material">
        <xsd:annotation/>
        <xsd:complexType>
```

```
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="Issue_Carbon_Dioxide_CertificateInput">
    <wsdl:part element="tns:Individual_Eco_Value" name="Calculated_Eco_Value"/>
  </wsdl:message>
  <wsdl:message name="Calculate_Carbon_DioxideInput">
    <wsdl:part element="tns:Material" name="Available_Material"/>
  </wsdl:message>
  <wsdl:message name="Issue_Carbon_Dioxide_CertificateOutput">
    <wsdl:part element="tns:Certificate" name="Issued_Certificate"/>
  </wsdl:message>
  <wsdl:message name="Calculate_Carbon_DioxideOutput">
    <wsdl:part element="tns:Individual_Eco_Value" name="Calculated_Eco_Value"/>
  </wsdl:message>
  <wsdl:portType name="Eco_Value_Calculation">
    <wsdl:operation name="Calculate_Carbon_Dioxide">
      <wsdl:documentation>Calculates the carbon dioxide for any given material</wsdl:documentation>
      <wsdl:input message="tns:Calculate_Carbon_DioxideInput"/>
      <wsdl:output message="tns:Calculate_Carbon_DioxideOutput"/>
    </wsdl:operation>
    <wsdl:operation name="Issue_Carbon_Dioxide_Certificate">
      <wsdl:documentation>Issues Certificate</wsdl:documentation>
      <wsdl:input message="tns:Issue_Carbon_Dioxide_CertificateInput"/>
      <wsdl:output message="tns:Issue_Carbon_Dioxide_CertificateOutput"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Entrepreneur Insurance Bundle Service

Listing A.11: Business Service Diagram XML for Entrepreneur Insurance Bundle

```
<?xml version="1.0" encoding="UTF-8"?>
<bsm:ValueModel xmi:version="2.0" xmlns:xmi="http://www.omg.org
      /XMI" xmlns:bsm="http://www.siemens.com/CT/texo/ise/
      BusinessServiceModel" author="Gregor␣Scheithauer" created="
      2010-04-15" version="1">
  <RevenueModel revenueModelID="_YuqKQEiREd-CfbCAB98smw" name="
        for␣Entrepreneurs" streamType="Transaction␣Cut"
        pricingMethod="Market-based␣Price" RM_TC="_T7sxkEiREd-
        CfbCAB98smw"/>
  <TargetCustomer targetCustomerID="_T7sxkEiREd-CfbCAB98smw"
        name="Entrepreneurs">
    <consistsOf customerEquity="Acquisition"/>
  </TargetCustomer>
  <ValueObject valueObjectID="_HfhkwEiREd-CfbCAB98smw" name="
        Mandate" type="Tangible"/>
  <ValueObject valueObjectID="_KyrXkEiREd-CfbCAB98smw" name="
        Recommendation" type="Tangible"/>
  <ValueObject valueObjectID="_2bVZEEiSEd-CfbCAB98smw" name="
        Low␣Transaction␣Costs" type="Intangible"/>
  <ValueObject valueObjectID="_4YgQYEiSEd-CfbCAB98smw" name="
        Ongoing␣Consultancy" type="Intangible"/>
  <ValueObject valueObjectID="_7BoOwEiSEd-CfbCAB98smw" name="
        Individual␣Consultancy" type="Intangible"/>
  <DistributionChannel distributionChannelID="_cZgEUEiREd-
        CfbCAB98smw" name="TEXO␣Service␣Marketplace"
        customerBuyingCycle="Awareness"/>
  <ValueOffer name="Entrepreneur␣Insurance␣Bundle" reasoning="
        Risk␣Reduction" valueLevel="Commodity" priceLevel="Market
        " lifeCycleStep="Value␣Use" VOf_DC="_cZgEUEiREd-
        CfbCAB98smw" VOf_RM="_YuqKQEiREd-CfbCAB98smw" VOf_TC="
        _T7sxkEiREd-CfbCAB98smw" VOf_VOb="_HfhkwEiREd-CfbCAB98smw
        ␣_KyrXkEiREd-CfbCAB98smw␣_2bVZEEiSEd-CfbCAB98smw␣
        _4YgQYEiSEd-CfbCAB98smw␣_7BoOwEiSEd-CfbCAB98smw"/>
</bsm:ValueModel>
```

Listing A.12: Conceputal Service Diagram XML for Entrepreneur Insurance Bundle

```
<?xml version="1.0" encoding="UTF-8"?>
<csm:CSM xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:csm="http://www.siemens.com/CT/texo/ise/
      ConceptualServiceModel">
  <CSM_SProduct name="Entrepreneur␣Insurance␣Bundle" key="
        EIB00001" description="Individual␣recommendations␣for␣a␣
        complete␣and␣individualized␣insurance␣packet␣for␣
        entrepreneurs" version="2" created="2010-02-22T00:00:00
        .000+0100" updated="2010-02-23T00:00:00.000+0100"
        automation="Partially␣Automated" composition="Final␣
        Service" customizable="true">
    <spokenLanguage>ger</spokenLanguage>
    <writtenLanguage>ger</writtenLanguage>
    <nextUpdate>2011-11-24T00:00:00.000+0100</nextUpdate>
    <SProduct_Payment name="Bank␣Transfer" instrument="Bank␣
          Transfer" preferred="true" recurrence="Monthly"/>
    <SProduct_Certificate provider="IHK" title="
          Erlaubnisurkunde" created="2006-06-06T00:00:00.000+0200
          "/>
    <SProduct_Certificate provider="CRM␣Company" title="CRM␣
          Product␣Expert" created="2007-09-01T00:00:00.000+0200"/
          >
    <SProduct_Standard provider="ISO" title="ISO␣9001:2008"
          status="N/A" created="2008-12-01T00:00:00.000+0100"
          author="ISO" version="Dec␣2008"/>
    <SProduct_Classification value="360055" system="Nice␣
          Classification"/>
    <SProduct_Classification value="524210"/>
    <SProduct_TermsOfUse paymentCondition="http://insurance-
          broker.com/paymentcondition" deliveryCondition="http://
          insurance-broker.com/deliverycondition"/>
    <SProduct_Benefit name="Individual␣Consulting"/>
    <SProduct_Benefit name="Ongoing␣Consulting"/>
    <SProduct_Benefit name="Low␣Transaction␣Costs"/>
    <SProduct_Actor xsi:type="csm:Partner" name="CCI" key="
          ACT_KEY_CCI" DUNS="N/A" industry="Public␣Sector"/>
    <SProduct_Actor xsi:type="csm:Partner" name="Insurer␣B" key
          ="ACT_KEY_IB" DUNS="N/A" industry="Insurance"/>
    <SProduct_Actor xsi:type="csm:Partner" name="Insurer␣A" key
          ="ACT_KEY_IA" DUNS="N/A" industry="Insurance"/>
    <SProduct_Actor xsi:type="csm:Partner" name="Underwriter"
          key="ACT_KEY_UWA" DUNS="N/A" industry="Insurance"/>
    <SProduct_Actor xsi:type="csm:Provider" name="Insurance␣Broker" key="ACT_KEY_IS" DUNS="N/A" industry="Insurance"/>
    <SProduct_Resource name="Broker␣Mandate" type="Capability"/>
    <SProduct_Resource name="Company␣Data" type="Information"/>
    <SProduct_Resource name="Entrepreneur␣Data" type="Information"/>
    <SProduct_Resource name="Customer␣Risk␣Rating" type="Information"/>
    <SProduct_Resource name="Commercial␣Object" type="Information"/>
    <SProduct_Resource name="Recommendation" type="Information"/>
    <SProduct_Resource name="Contract" type="Information"/>
    <SProduct_Quality name="Recommendation">
      <Quality_Performance capacity="20">
        <Performance_Latency value="4" granularity="Hour"/>
        <Performance_Throughput events="3" recurrence="Weekly"/>
      </Quality_Performance>
      <Quality_Dependability availability="0.9" reliability="3" accuracy="1"/>
      <Quality_Security authentication="true">
        <Security_Confidentiality encrypted="true" encryptType="N/A"/>
      </Quality_Security>
    </SProduct_Quality>
    <SProduct_Price xsi:type="csm:TransactionCut" name="Recommendation"/>
    <SProduct_Channel name="Startup␣Service␣Ecosystem"/>
    <SProduct_Capability name="Recommendation␣for␣Insurance␣Bundle" interface="Technical␣Interface" duration="2" durationGran="Day" Capability_ref_Price="//@CSM_SProduct.0/@SProduct_Price.0" Capability_ref_Quality="//@CSM_SProduct.0/@SProduct_Quality.0">
      <Capability_Condition xsi:type="csm:PreCondition" name="Signed␣Broker␣Mandate" state="signed" Condition_ref_Resource="//@CSM_SProduct.0/@SProduct_Resource.0"/>
      <Capability_Condition xsi:type="csm:PreCondition" name="Available␣Company␣Data" state="available" Condition_ref_Resource="//@CSM_SProduct.0/@SProduct_Resource.1"/>
      <Capability_Condition xsi:type="csm:PreCondition" name="
```

```
            Available␣Risk␣Rating" state="available"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.3"/>
        <Capability_Condition xsi:type="csm:PreCondition" name="
            Available␣Comm.␣Obj." state="available"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.4"/>
        <Capability_Condition xsi:type="csm:PreCondition" name="
            Available␣Entrepreneur␣Data" description="" state="
            available" Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.2"/>
        <Capability_Condition xsi:type="csm:PostCondition" name="
            Created␣Recommendation" state="created"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.5"/>
        <Capability_Condition xsi:type="csm:PostCondition" name="
            Signed␣Contract" state="signed"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.6"/>
      </SProduct_Capability>
      <SProduct_Capability name="Consultancy" interface="
          Technical␣Interface" duration="30" durationGran="Minute
          ">
        <Capability_Condition xsi:type="csm:PreCondition" name="
            Signed␣Contract" state="signed"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.6"/>
      </SProduct_Capability>
    </CSM_SProduct>
  </csm:CSM>
```

Listing A.13: WSDL for Entrepreneur Insurance Bundle

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:tns="http://www.example.org/NewWSDLFile/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema" name="
    Entrepreneur_Insurance_Bundle" targetNamespace="http://www.
    example.org/NewWSDLFile/">
  <wsdl:documentation>[key:EIB00001][description:Individual
      recommendations for a complete and individualized
      insurance packet for entrepreneurs][spkn 1.:ger;][wr 1.
      :ger;][version:2][created:Mon Feb 22 00:00:00 CET 2010][
      updated:Tue Feb 23 00:00:00 CET 2010][next update:Thu Nov
       24 00:00:00 CET 2011;][type:Core Service][
      automation:Partially Automated][composition:Final Service
      ][customizable:true][paym. cond.:http://insurance-broker.
      com/paymentcondition][deli. cond.:http://insurance-broker
      .com/deliverycondition][classi.:360055 Nice
      Classification][classi.:524210 NAICS][benefit:Individual
      Consulting][benefit:Ongoing Consulting][benefit:Low
      Transaction Costs]</wsdl:documentation>
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.example.org/
        NewWSDLFile/">
      <xsd:element name="Broker_Mandate">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Contract">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Recommendation">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Customer_Risk_Rating">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Company_Data">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Commercial_Object">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Entrepreneur_Data">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="Recommendation_for_Insurance_BundleInput"
      >
    <wsdl:part element="tns:Broker_Mandate" name="
        Signed_Broker_Mandate"/>
    <wsdl:part element="tns:Company_Data" name="
        Available_Company_Data"/>
    <wsdl:part element="tns:Customer_Risk_Rating" name="
        Available_Risk_Rating"/>
    <wsdl:part element="tns:Commercial_Object" name="
        Available_Comm._Obj."/>
    <wsdl:part element="tns:Entrepreneur_Data" name="
        Available_Entrepreneur_Data"/>
  </wsdl:message>
  <wsdl:message name="ConsultancyInput">
    <wsdl:part element="tns:Contract" name="Signed_Contract"/>
  </wsdl:message>
  <wsdl:message name="Recommendation_for_Insurance_BundleOutput
      ">
    <wsdl:part element="tns:Recommendation" name="
        Created_Recommendation"/>
    <wsdl:part element="tns:Contract" name="Signed_Contract"/>
  </wsdl:message>
  <wsdl:portType name="Entrepreneur_Insurance_Bundle">
    <wsdl:operation name="Recommendation_for_Insurance_Bundle">
      <wsdl:documentation>Development of an individual
          recommendation for a set of insurance policies.</
          wsdl:documentation>
      <wsdl:input message="
          tns:Recommendation_for_Insurance_BundleInput"/>
      <wsdl:output message="
          tns:Recommendation_for_Insurance_BundleOutput"/>
    </wsdl:operation>
    <wsdl:operation name="Consultancy">
      <wsdl:documentation>Ongoing and individual consultancy.</
          wsdl:documentation>
      <wsdl:input message="tns:ConsultancyInput"/>
    </wsdl:operation>
  </wsdl:portType>
```

```
</wsdl:definitions>
```

Manage Client Hardware Service

Listing A.14: Business Service Diagram XML for Manage Client Hardware

```
<?xml version="1.0" encoding="UTF-8"?>
<bsm:ValueModel xmi:version="2.0" xmlns:xmi="http://www.omg.org
      /XMI" xmlns:bsm="http://www.siemens.com/CT/texo/ise/
      BusinessServiceModel" author="Gregor␣Scheithauer" created="
      2010-04-15" version="1">
  <RevenueModel revenueModelID="_YuqKQEiREd-CfbCAB98smw" name="
        for␣Entrepreneurs" streamType="Transaction␣Cut"
        pricingMethod="Market-based␣Price" RM_TC="_T7sxkEiREd-
        CfbCAB98smw"/>
  <TargetCustomer targetCustomerID="_T7sxkEiREd-CfbCAB98smw"
        name="Entrepreneurs">
    <consistsOf customerEquity="Acquisition"/>
  </TargetCustomer>
  <ValueObject valueObjectID="_HfhkwEiREd-CfbCAB98smw" name="
        Hardware" type="Tangible"/>
  <ValueObject valueObjectID="_KyrXkEiREd-CfbCAB98smw" name="
        Low␣Transaction␣Costs" type="Intangible"/>
  <ValueObject valueObjectID="_4YgQYEiSEd-CfbCAB98smw" name="
        Low␣Labor␣Costs" type="Intangible"/>
  <ValueObject valueObjectID="_7BoOwEiSEd-CfbCAB98smw" name="
        Low␣IT␣Costs" type="Intangible"/>
  <ValueObject valueObjectID="_Vu8l8EiTEd-CfbCAB98smw" name="
        Recent␣Hardware" type="Intangible"/>
  <DistributionChannel distributionChannelID="_cZgEUEiREd-
        CfbCAB98smw" name="TEXO␣Service␣Marketplace"
        customerBuyingCycle="Awareness"/>
  <ValueOffer name="Entrepreneur␣Insurance␣Bundle" reasoning="
        Risk␣Reduction" valueLevel="Commodity" priceLevel="Market
        " lifeCycleStep="Value␣Use" VOf_DC="_cZgEUEiREd-
        CfbCAB98smw" VOf_RM="_YuqKQEiREd-CfbCAB98smw" VOf_TC="
        _T7sxkEiREd-CfbCAB98smw" VOf_VOb="_HfhkwEiREd-CfbCAB98smw
        ␣_KyrXkEiREd-CfbCAB98smw␣_4YgQYEiSEd-CfbCAB98smw␣
        _7BoOwEiSEd-CfbCAB98smw␣_Vu8l8EiTEd-CfbCAB98smw"/>
</bsm:ValueModel>
```

Listing A.15: Conceputal Service Diagram XML for Manage Client Hardware

```
<?xml version="1.0" encoding="UTF-8"?>
<csm:CSM xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:csm="http://www.siemens.com/CT/texo/ise/
    ConceptualServiceModel">
  <CSM_SProduct name="Manage␣Client␣Hardware" key="http://www.
      itcompany.com/mch" description="allow␣outsourcing␣of␣
      purchasing␣and␣the␣maintaining␣of␣computer␣hardware"
      documentation="www.documentation.de" version="1" created=
      "2009-10-06T00:00:00.000+0200" updated="2010-03-01
      T00:00:00.000+0100" automation="Partially␣Automated"
      composition="Final␣Service">
    <spokenLanguage>eng</spokenLanguage>
    <spokenLanguage>ger</spokenLanguage>
    <writtenLanguage>eng</writtenLanguage>
    <writtenLanguage>ger</writtenLanguage>
    <nextUpdate>2011-11-10T00:00:00.000+0100</nextUpdate>
    <SProduct_Payment name="Bank␣Transfer" description=""
        instrument="Bank␣Transfer" preferred="true" recurrence=
        "Monthly"/>
    <SProduct_Discount xsi:type="csm:PaymentDiscount" name="
        Bank␣Transfer␣Disc." allowance="0.01"
        PDiscount_ref_Payment="//@CSM_SProduct.0/
        @SProduct_Payment.0"/>
    <SProduct_Discount xsi:type="csm:SeasonalDiscount" name="
        New␣Fiscal␣Year␣Disc." allowance="0.02" from="
        2009-12-01T00:00:00.000+0100" to="2010-01-31T00:00:00
        .000+0100"/>
    <SProduct_Rating name="Rating␣1" created="2010-01-05
        T00:00:00.000+0100" comment="fast␣service">
      <servqual>Responsiveness</servqual>
      <eightPs>Productivity &amp; Quality</eightPs>
    </SProduct_Rating>
    <SProduct_Classification value="350097"/>
    <SProduct_TermsOfUse paymentCondition="http://itc.com/
        services/mch/paymentcondition" deliveryCondition="http:
        //itc.com/services/mch/deliverycondition"/>
    <SProduct_Benefit name="Low␣Transaction␣Costs"/>
    <SProduct_Benefit name="Low␣Labor␣Costs"/>
    <SProduct_Benefit name="Low␣IT␣Costs"/>
    <SProduct_Benefit name="Recent␣Hardware"/>
    <SProduct_Actor xsi:type="csm:Provider" name="IT␣Company"
        key="ACT_KEY_ITC" DUNS="123456789" industry="IT␣
```

```
          Services">
      <Actor_Contact personName="Renate␣Schmitz" phone="12␣3456
           ␣789" email="rs@it.com"/>
    </SProduct_Actor>
    <SProduct_Resource name="Hardware"/>
    <SProduct_Resource name="Order" type="Information"/>
    <SProduct_Quality name="New␣HW␣Quali.">
      <Quality_Performance capacity="5">
        <Performance_Latency value="2" granularity="Second"/>
        <Performance_Throughput events="5" recurrence="Dayly"/>
      </Quality_Performance>
      <Quality_Dependability availability="0.875" reliability="
           7" reliabilityGran="Day" maintainability="1"
           maintainaGran="Day" accuracy="1"/>
      <Quality_Security authentication="true" authorization="
           true">
        <Security_Confidentiality encrypted="true" keyLength="
             160" encryptType="ECRYPT␣II"/>
        <Security_DIntegrity value="50"/>
      </Quality_Security>
    </SProduct_Quality>
    <SProduct_Price xsi:type="csm:UsageBased" name="Per␣new␣
         Hardware" amount="500.0" tax="0.19"/>
    <SProduct_Channel name="Intranet" description="this␣is␣a␣
         Channel" productVariety="20" waitingTime="48"
         waitingTimeGran="Hour" type="Electronically"/>
    <SProduct_Capability name="Return␣Hardware" interface="Web␣
         Interface" duration="24" durationGran="Hour">
      <Capability_Condition xsi:type="csm:PreCondition" name="
           New␣Order" state="new" Condition_ref_Resource="//
           @CSM_SProduct.0/@SProduct_Resource.1"/>
      <Capability_Condition xsi:type="csm:PostCondition" name="
           Returned␣Hardware" state="returned"
           Condition_ref_Resource="//@CSM_SProduct.0/
           @SProduct_Resource.0"/>
    </SProduct_Capability>
    <SProduct_Capability name="Order␣New␣Hardware" description=
         "this␣is␣a␣new␣Hardware" interface="Web␣Interface"
         duration="24" durationGran="Hour" Capability_ref_Price=
         "//@CSM_SProduct.0/@SProduct_Price.0"
         Capability_ref_Quality="//@CSM_SProduct.0/
         @SProduct_Quality.0">
      <Capability_Condition xsi:type="csm:PreCondition" name="
           New␣Order" state="new" Condition_ref_Resource="//
           @CSM_SProduct.0/@SProduct_Resource.1"/>
      <Capability_Condition xsi:type="csm:PostCondition" name="
```

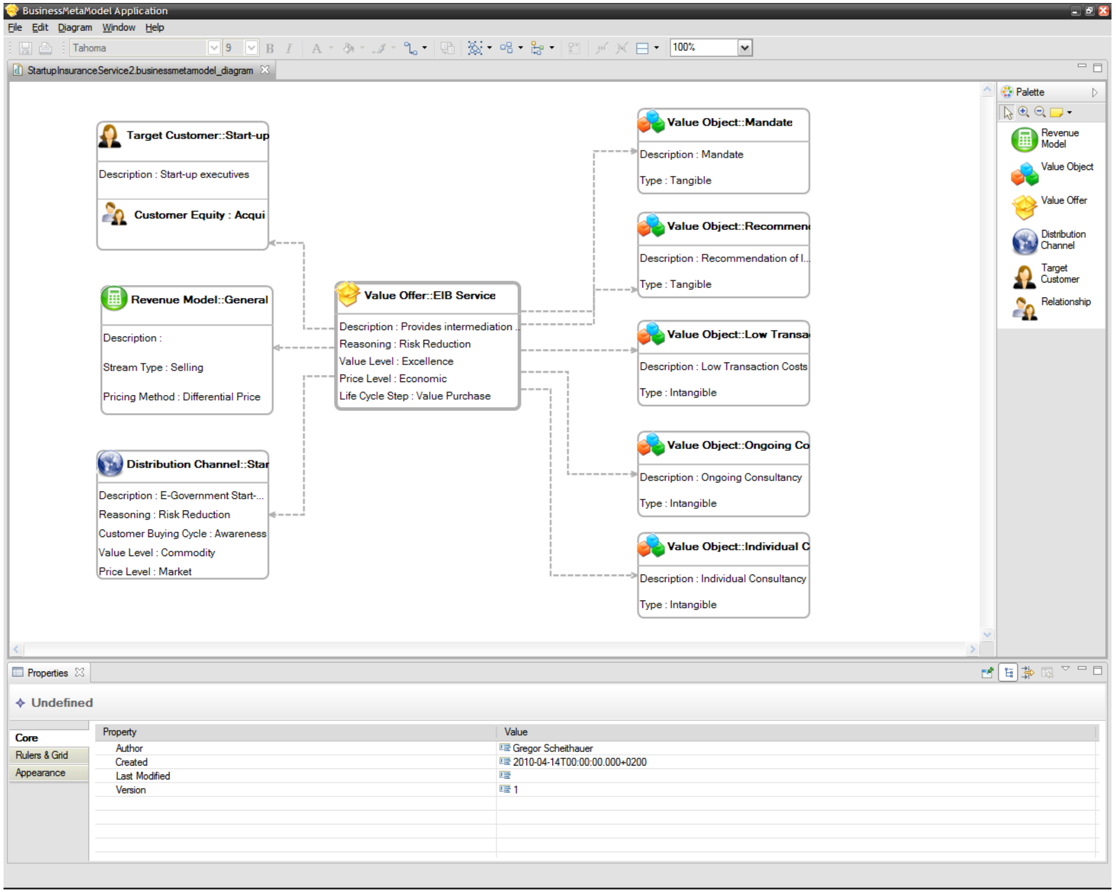
```
            New␣Hardware" description="" state="new"
            Condition_ref_Resource="//@CSM_SProduct.0/
            @SProduct_Resource.0"/>
    </SProduct_Capability>
  </CSM_SProduct>
</csm:CSM>
```

Listing A.16: WSDL for Manage Client Hardware

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:tns="http://www.example.org/NewWSDLFile/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema" name="
    Manage_Client_Hardware" targetNamespace="http://www.example
    .org/NewWSDLFile/">
  <wsdl:documentation>[key:http://www.itcompany.com/mch][
      description:allow outsourcing of purchasing and the
      maintaining of computer hardware][documentation:www.
      documentation.de][spkn l.:eng;ger;][wr l.:eng;ger;][
      version:1][created:Tue Oct 06 00:00:00 CEST 2009][
      updated:Mon Mar 01 00:00:00 CET 2010][next update:Thu Nov
       10 00:00:00 CET 2011;][type:Core Service][
      automation:Partially Automated][composition:Final Service
      ][customizable:false][paym. cond.:http://itc.com/services
      /mch/paymentcondition][deli. cond.:http://itc.com/
      services/mch/deliverycondition][classi.:350097 NAICS][
      benefit:Low Transaction Costs][benefit:Low Labor Costs][
      benefit:Low IT Costs][benefit:Recent Hardware]</
      wsdl:documentation>
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.example.org/
        NewWSDLFile/">
      <xsd:element name="Order">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Hardware">
        <xsd:annotation/>
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
```

```xml
  </wsdl:types>
  <wsdl:message name="Order_New_HardwareInput">
    <wsdl:part element="tns:Order" name="New_Order"/>
  </wsdl:message>
  <wsdl:message name="Return_HardwareInput">
    <wsdl:part element="tns:Order" name="New_Order"/>
  </wsdl:message>
  <wsdl:message name="Return_HardwareOutput">
    <wsdl:part element="tns:Hardware" name="Returned_Hardware"/>
  </wsdl:message>
  <wsdl:message name="Order_New_HardwareOutput">
    <wsdl:part element="tns:Hardware" name="New_Hardware"/>
  </wsdl:message>
  <wsdl:portType name="Manage_Client_Hardware">
    <wsdl:operation name="Return_Hardware">
      <wsdl:documentation>Allows to return once ordered hardware.</wsdl:documentation>
      <wsdl:input message="tns:Return_HardwareInput"/>
      <wsdl:output message="tns:Return_HardwareOutput"/>
    </wsdl:operation>
    <wsdl:operation name="Order_New_Hardware">
      <wsdl:documentation>Form to order new hardware.</wsdl:documentation>
      <wsdl:input message="tns:Order_New_HardwareInput"/>
      <wsdl:output message="tns:Order_New_HardwareOutput"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

B. Screenshots



Target Customer::Start-up
Description : Start-up executives
Customer Equity : Acqui

Revenue Model::General
Description :
Stream Type : Selling
Pricing Method : Differential Price

Distribution Channel::Star
Description : E-Government Start-...
Reasoning : Risk Reduction
Customer Buying Cycle : Awareness
Value Level : Commodity
Price Level : Market

Value Offer::EIB Service
Description : Provides intermediation ..
Reasoning : Risk Reduction
Value Level : Excellence
Price Level : Economic
Life Cycle Step : Value Purchase

Value Object::Mandate
Description : Mandate
Type : Tangible

Value Object::Recommen
Description : Recommendation of I...
Type : Tangible

Value Object::Low Transa
Description : Low Transaction Costs
Type : Intangible

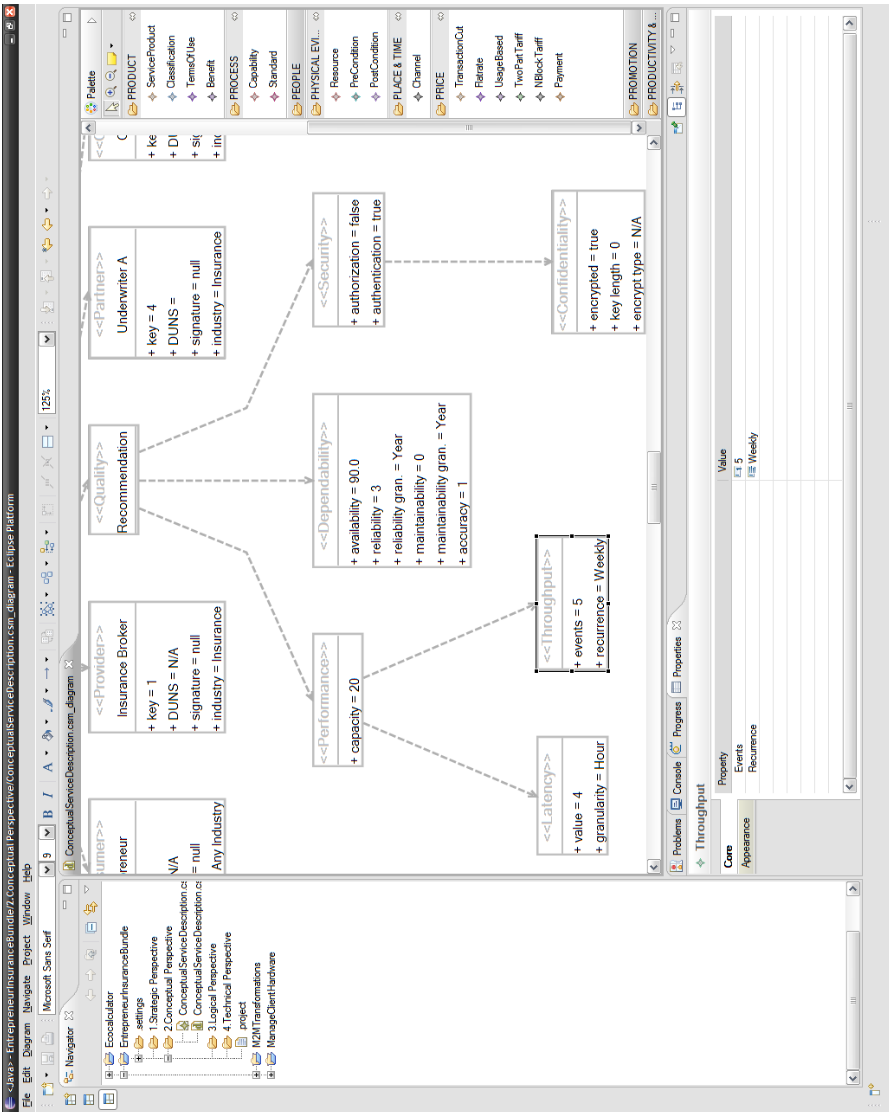
Value Object::Ongoing Co
Description : Ongoing Consultancy
Type : Intangible

Value Object::Individual C
Description : Individual Consultancy
Type : Intangible

Undefined

Property	Value
Author	Gregor Scheithauer
Created	2010-04-14T00:00:00.000+0200
Last Modified	
Version	1

Figure B.1.: Business Service Modeling Tool Screenshot



Property	Value
Events	5
Recurrence	Weekly

Figure B.2.: Conceptual Service Modeling Tool Screenshot

UNIVERSITY OF BAMBERG PRESS

Im Rahmen der fortschreitenden Globalisierung und des technischen Wandels spielt das Internet eine immer größere Rolle beim Anbieten und Vermitteln von Dienstleistungen. Zur selben Zeit spezialisieren sich Unternehmen zunehmend auf ihre Kernkompetenzen und schließen sich in „Service Ecosystems“ zusammen, um flexibel auf den Markt reagieren zu können. Eine wichtige Fragestellung hierbei ist wie neue Dienstleistungen innerhalb von Service Ecosystems entwickelt und beschrieben werden können, um effizient über das Internet gehandelt zu werden. Hierzu schlägt die vorliegende Arbeit eine Methode zur Beschreibung von Dienstleistungen vor, die sich in Dienstleistungsentwicklungsprozesse integrieren lässt.

Die Entwicklung einer solchen Methode führt zu drei Herausforderungen: Erstens muss herausgefunden werden welche Eigenschaften sich für die Beschreibung von Dienstleistungen eignen. Diese Arbeit untersucht existierende Ansätze im Bereich Marketing, Wirtschaftsinformatik und Angewandte Informatik und erarbeitet ein Modell für eine formale Beschreibung von Dienstleistungen, welches das Anbieten und Finden von Dienstleistungen im Internet vereinfacht. Zweitens muss die Aufnahme, Dokumentation und Kommunikation von Beschreibungen über den gesamten Dienstleistungsentwicklungsprozess gewährleistet werden. Der in der Arbeit verfolgte Ansatz ist die Entwicklung einer geeigneten Modellierungsnotation als Erweiterung der Unified Modeling Language (UML). Drittens bedarf es einer Überführung der konzeptionellen Beschreibung für Dienstleistungen in Softwarerealisierungssprachen, die im Internet Anwendung finden. Der Beitrag dieser Arbeit umfasst die automatische Überführung der Dienstleistungsbeschreibung in ein Web Services Description Language (WSDL)-Dokument mittels Modell-zu-Modell-Transformationen.

Die Methode zur Beschreibung von Dienstleistungen wurde einerseits durch Implementierung einer integrierten Modellierungsumgebung und zugehöriger Transformationsskripte und andererseits durch zwei Fallstudien in der Versicherungs- und IT-Outsourcing-

ISBN 978-3-86309-007-4
ISSN 1867-7401
21,50 Euro