

UNIVERSITY OF OSLO
Department of Informatics

BPMN4SOA

A service oriented
process modelling
language

Master thesis
60 credits

Eivind Bergstøl

10th May 2010



Acknowledgements

The process of writing this thesis had not been possible without the guidance of my two supervisors **Dr. Arne-Jørgen Berre** (Chief Scientist, SINTEF ICT) and **Brian Elvesæter** (Research Scientist/Research Fellow, SINTEF ICT) and I want to express my sincere thanks to them both. The oversight that Arne-Jørgen has in this field is mindboggling, and as a supervisor he possesses the ability to say just the right thing to adjust the course of my work. Brian has been my “go to guy” and is always available for a good discussion and he provides insight to most topics in the modelling domain. I would also like to add that being the group teacher in INF5120, a UIO course taught by Arne-Jørgen Berre, has helped me a great deal in reasoning on SoaML and BPMN 2.0.

I would also express my sincere thanks to all of my friends for being there for me the last year keeping my spirits up and making me believe in myself, and in particular, my brother **Tormod Bergstøl** and father **Svein Bergstøl** and especially my mother **Eli Bergstøl** whom I suddenly lost to cancer while working on this thesis.

Oslo, 10th May 2010

Abstract

Service oriented architectures have become very popular the last few years. The abstraction of computer systems into a service paradigm bring many new solutions, both for cross business processes to aid interoperability and the reuse of existing legacy systems in a new network centric world.

In the wake of this, service modelling has become a part of OMGs Model Driven Architecture and new modelling languages that are based on past experience for the new paradigm are emerging. BPMN 2.0 and SoaML are the newest modelling standards from OMG that focus on service modelling. They provide different approaches to the service domain where BPMN 2.0 emphasise process modelling and SoaML emphasise service architecture modelling.

BPMN4SOA is a language that extends the use of BPMN 2.0, and bring more emphasis on the service modelling capability of BPMN 2.0. It does this by means of role modelling to abstract the participants and service choreographies into reusable objects. BPMN4SOA also provide modelling capability for information data for messages through implementation of UML at L0 compliance.

Because BPMN4SOA is an extension of BPMN 2.0 through it's Extension and External Relationship constructions, BPMN4SOA should be implementable in all systems fully compliant with BPMN 2.0 specification.

Contents

Acknowledgements	I
Abstract	III
Contents	V
List of figures	IX
List of tables	XII
1 Introduction	1
1.1 Background	1
1.2 Research questions	2
1.3 Structure of this thesis	2
2 Research method and scope	5
2.1 Research method	5
2.2 Scope	6
2.3 Outlining the task	8
2.4 Tool support	9
3 Case and domain description	11
3.1 Why this case	11
3.2 The Travel booking case - outline	12
3.3 Business Process Management (BPM)	13
3.4 The domain of modeling	13
3.4.1 Objects and the object paradigm	14
3.4.2 Role modelling	15
3.4.3 Services and service oriented architectures (SOA)	16
3.5 Model Driven Engineering	17
3.5.1 Model Driven Architecture (MDA)	18

3.5.2	MDD	18
4	Requirements for process, information and service modelling	21
4.1	Descriptions from specification	21
4.1.1	Short description SoaML	21
4.1.2	Short description BPMN 2.0	22
4.2	Zachman Framework	22
4.3	Requirements for a service oriented process modelling language	23
4.3.1	Requirements for service modelling	24
4.3.2	Requirements for information modelling	25
4.3.3	Requirements for process modelling	25
4.4	Scoring of requirements	27
5	Evaluation of BPMN 2.0 and SoaML	29
5.1	BPMN 2.0	29
5.1.1	Usage	30
5.1.2	Constructions and their meta model	33
5.1.2.1	Extensibility	33
5.1.2.2	External Relationships	34
5.1.2.3	Conversation Assosiation	34
5.1.2.4	Interaction Specification	34
5.1.2.5	MessageFlow	35
5.1.2.6	Participant	35
5.1.2.7	Services	36
5.1.2.8	Tasks	36
5.1.2.9	ServiceTask	36
5.1.2.10	SendTask and ReceiveTask	36
5.1.2.11	Conversation diagram	36
5.1.2.12	Choreography diagram	37
5.1.3	Travel case - BPMN 2.0	37
5.1.4	Requirement evaluation	40
5.1.4.1	Service requirement evaluation	40
5.1.4.2	Information requirement evaluation	43
5.1.4.3	Process requirement evaluation	44
5.2	SoaML	46
5.2.1	Usage	47
5.2.2	Travel case - SoaML	48
5.2.2.1	ServiceContract based SoaML	49
5.2.2.2	ServiceInterface based SoaML	51
5.2.2.3	Capabilities	53
5.2.3	Requirement evaluation	53

5.2.3.1	Service requirement evaluation	53
5.2.3.2	Information requirement evaluation	54
5.2.3.3	Process requirement evaluation	55
5.3	Conclusions based on the two evaluations	57
6	BPMN4SOA vision	59
6.1	Mapping and integration of SoaML to BPMN 2.0 notation	60
6.2	SoaML ServicesArchitecture	61
6.2.1	Participants	61
6.2.1.1	Service Architectures vs. Conversations	62
6.2.1.2	ServiceContract and ChoreographyTask	62
6.3	SoaML Participant Architecture	64
6.3.1	Service Interfaces	64
6.3.2	Capabilities	66
6.3.3	MessageType	66
6.4	Evaluation of mapping between BPMN 2.0 and SoaML	67
6.5	Mapping conclusions	70
7	The BPMN4SOA specification	73
7.1	Background	73
7.2	Role conversation diagram	74
7.3	Collaboration constraints	76
7.4	BPMN4SOA information modelling	76
7.5	Conclusion	77
8	BPMN4SOA tool implementation	79
8.1	Eclipse EMF/GMF and Eclipse for SOA	79
8.2	BPMN4SOA tool	81
8.3	Usage and evaluation	83
9	Evaluation of BPMN4SOA	85
9.1	BPMN4SOA travel case	85
9.2	Requirement evaluation	86
9.2.1	Service requirement evaluation	86
9.2.2	Information requirement evaluation	87
9.2.3	Process requirement evaluation	87
9.3	Conclusions of evaluation	88
10	Conclusion and future work	89
10.1	Conclusion of requirements evaluation	89
10.2	Future work	91

References	93
Appendices	97
A OOram - Role Modeling	99
A.1 Collaboration view	100
A.2 Interface view	102
A.3 Scenario view	102
B Selected BPMN 2.0 meta models	103
C Athena CBP with BPMN 2.0	111
C.1 Background	111
C.2 Problem and research questions	111
C.3 Method	112
C.4 Scenario	112
C.5 ATHENA CPB	112
C.5.1 The dimensions	112
C.5.2 Work methods	114
C.5.3 MO2GO and IEM in CBP	115
C.6 BPMN 2.0	116
C.6.1 Collaborations	116
C.6.2 Conversations	116
C.6.3 Choreographies	116
C.7 Discussion	118
C.8 Conclusion	118
D Terms, definitions and abbreviations	119
D.1 eXtensible Markup Language (XML)	119
D.2 XML Metadata Interchange (XMI)	119
D.3 Service Oriented Architecture (SOA)	119
D.4 Object Oriented Role Analysis Method (OORAM)	120
D.5 Object Management Group (OMG)	120
D.6 Model Driven Architecture (MDA)	120
D.7 Meta-Object Facility (MOF)	120
D.8 Computation Independent Model (CIM)	120
D.9 Platform Independent Model (PIM)	121
D.10 Platform Specific Model (PSM)	121
D.11 Unified Modeling Language (UML)	121

List of Figures

2.1	Screenshot from Signavio and Modelio.	10
3.1	Model-driven development (MDD) lifeline	19
4.1	View of the Zachman Framework	23
5.1	Example of a simple private process based on a travel case created for documentation purposes. One is created outside a participant, the other inside the participant (Pool).	30
5.2	A public process showing the MessageFlows that exists between the traveller and the process.	31
5.3	A public process showing the message flows that exists between the traveller and the process.	31
5.4	A simple Choreography between a traveller and a hotel initiated by the traveller	32
5.5	A simple Conversation between a traveller and a hotel	32
5.6	A representation of BPMN Core and Layer Structure	33
5.7	Travel case Conversation	38
5.8	Choreography between traveller and BookingSite to book hotel and flight	38
5.9	Choreography between booking site and hotel/airline.	38
5.10	A Choreography between all participants detailing the payment process	39
5.11	A BPMN 2.0 Process detailing the choreography in figure 5.8 on page 38 and 5.9 on page 38	40
5.12	BPMN Process detailing the choreography in figure 5.10 on page 39 for payment and booking confirmations	41

5.13	Participating in Service - The use of conjugate (tilde) naming for opposite service interfaces	48
5.14	SoaML services architecture view of the travel case	49
5.15	SoaML service contract view of the travel case	49
5.16	UML interaction diagram detailing a protocol between an online-Booker and an onlineBookingProvider	50
5.17	SoaML ServiceInterfaces for the payment part.	51
5.18	SoaML MessageTypes for the interfaces' operations	51
5.19	Two participant architectures with ServiceInterface ports	52
5.20	Two participant architectures with ServiceInterface ports	52
5.21	SoaML participant capability view with ServiceInterfaces	53
6.1	Process with two communications and choreographies representing one ServiceInterface and one simple interface	65
6.2	Model alternatives of a BPMN process that can be thought of as a SoaML ServiceInterface	65
6.3	Three BPMN models for the simplest possible SOA and with supporting SoaML models for defining the interfaces	68
6.4	A BPMN payment example consisting of three roles	71
7.2	Conversation diagram meta model extended with Role construct	74
7.1	Roles displayed two ways	74
7.3	BPMN4SOA Role Conversation - upper shows the conversation. - lower shows an exploded version	75
8.1	BPMN4SOA GMF Tool Registry model	81
8.2	BPMN4SOA GMF Node Mapping model	81
8.3	BPMN4SOA GMF Figure Gallery model	82
8.4	Screenshot of RoleConversation diagram editor	84
9.1	BPMN4SOA Role Conversation diagram for the travel case	86
10.1	Zachman framework coverage for BPMN4SOA	91
10.2	Specialisation of the role concept for future work	92
10.3	Example drawing of the the extensions to the role conversation diagram for future work.	92
A.1	Many-to-many relationship between type, object, role and class.	100
A.2	Collaboration view notation	101
A.3	Collaboration view example	101
A.4	Interface view example	102
A.5	Scenario view example	102

B.1	BPMN 2.0 ConversationAssosiation class diagram	103
B.2	BPMN 2.0 External Relationships	104
B.3	BPMN 2.0 Extensions class diagram	104
B.4	BPMN 2.0 Interaction Specification class diagram	105
B.5	BPMN 2.0 Message flow class diagram	105
B.6	BPMN 2.0 Participant class diagram	106
B.7	BPMN 2.0 Services class diagram	106
B.8	BPMN 2.0 Task class diagram	107
B.9	BPMN 2.0 ServiceTask class diagram	107
B.10	BPMN 2.0 SendTask and ReceiveTask class diagram	108
B.11	BPMN 2.0 Conversation diagram class diagram	108
B.12	BPMN 2.0 Choreography diagram class diagram	109
C.1	Modelling Framework	113
C.2	Modelling Procedures	114
C.3	BPMN 2.0 Collaboration example	115
C.4	BPMN 2.0 Conversation example	117
C.5	BPMN 2.0 Choreography example	117

List of Tables

2.1	Description of values for scope discussion	7
2.2	Diagrams and constructs and the scope for this thesis	7
2.3	Requirements and support for various modelling tools	10
4.1	Requirements for service modelling	24
4.2	Requirements for information modelling	25
4.3	Requirements for process modelling	26
5.1	Scoring of requirements for service modelling in BPMN 2.0	42
5.2	Scoring of requirements for information modelling in BPMN 2.0	44
5.3	Scoring of requirements for process modelling in BPMN 2.0	44
5.4	Scoring of requirements for service modelling in SoaML	54
5.5	Scoring of requirements for information modelling in SoaML scores	55
5.6	Scoring of requirements for process modelling in SoaML scores	55
5.7	Comparing requirement scores for BPMN 2.0 and SoaML	57
6.1	SoaML to BPMN mapping equivalent	63
9.1	Scoring of requirements for service modelling in BPMN4SOA	86
9.2	Scoring of requirements for information modelling in BPMN4SOA	87
9.3	Scoring of requirements for process modelling in BPMN4SOA	88
10.1	Comparing requirement scores for BPMN 2.0, SoaML and BPMN4SOA	90

Introduction

This thesis is a proposal for a service oriented process modelling language called BPMN4SOA. BPMN4SOA is based on BPMN 2.0 with some extensions to provide better service modelling capabilities. The specification of BPMN4SOA is based on research on BPMN 2.0 and SoaML done by case modelling and a mapping discussion.

1.1 Background

The use of Model Driven Architecture (MDA) is an expanding research- and standardization area. The invention of the Internet has brought new needs and technologies for business communication. The proprietary technologies have been widely used and caused problems in interoperability and interchange of data between business partners. By creating modelling standards for data representation, enterprise architectures, systems architectures, business processes, business motivation and many more, Object Management Group (OMG ¹) provides a joint venture for all technology driven businesses. The use of communication standards seem to greatly improve the interoperability and data interchange between systems and organisations.

OMG is due to release two new standards for their MDA portfolio, SoaML and BPMN 2.0. SoaML is a UML profile for Service Oriented Architecture (SOA) and provides two approaches to the domain, ServiceContract- and ServiceInterface based SOA (22). BPMN 2.0 is a new version of OMGs own BPMN 1.x business process modelling language. BPMN 2.0 provides a formal meta model, new extensions for process modelling and new diagram types. In addition BPMN 2.0 has a formal mapping to execution languages, e.g. WS-BPEL (20).

¹<http://www.omg.net/>

1.2 Research questions

The creation of a service oriented process modelling language has to be based on some experience. There are many earlier languages that provide insight to the modelling domain. However, SoaML and BPMN 2.0 are the newest from OMG in this area and is expected to provide the most comprehensive and cutting edge functionality.

Questions that will be illuminated:

- What do the languages cover in terms of capabilities?
- In which degree do the languages bring a solution to the problem domain?
- What are the strengths and weaknesses?
- In which areas do the languages' modelling capabilities overlap each other?
- Can overlapping concepts be combined into a new language?
- How can the languages be combined to provide a new language?

Can the BPMN graphical notation, e.g. the boxes and events etc, be used as a graphical notation for diagrams in SoaML. The SoaML activity diagram has many similarities with the BPMN collaboration diagram, but with fewer element types. We can test it out by using a subset of the SoaML meta model without modifications and implement this in EMG/GMF (7) to create an activity model editor. What modifications can be done to implement more of the BPMN ideas; different event types and human tasks are also a problem area. The Conversation diagram of BPMN 2.0 also has similarities with the SoaML Collaboration diagram, but brings fewer constructs to the model.

1.3 Structure of this thesis

This thesis is divided into three separate parts. Part 1 (chapter 2, 3 and 4) discusses the research of the languages, discussion on the domain and case and the requirements that the languages are evaluated on. Part 2 (chapter 5) is the evaluation and discussion of SoaML and BPMN 2.0. Lastly, part 3 (chapter 6, 7, 8 and 9) provides a suggested new language to solve some of the issues identified. Chapter 10 is the conclusion and future work. Appendices consists of meta model figures for BPMN 2.0 discussions and an article, written as a part of the master writing process, and a description of OOram (25).

Chapter 2 contains a description of the applied research method for this work on SoaML, BPMN 2.0 and BPMN4SOA, and the scope of this discussion. Issues with tools support that is experienced in the process of writing the thesis and the description of scoring of requirements are also covered.

Chapter 3 presents the case that all models are created around, and with a discussion on the domain of BPM, OO, role modelling and MDA.

Chapter 4 presents a short description of the defined domain that SoaML and BPMN 2.0 is meant to cover, and a discussion on the Zachman framework for enterprise architecture. Based on these discussions, a set of requirements for the languages (BPMN 2.0, SoaML and BPMN4SOA) are presented.

Chapter 5 contains the presentation, discussion and evaluation of BPMN 2.0 and SoaML.

Chapter 6 presents the idea of BPMN4SOA and a discussion of how to choose a foundation for the new language (Side by side, meta model mapping or meta model integration). It contains the discussion of a direct mapping between SoaML and BPMN 2.0 viewed from SoaML. Where and in what degree do they overlap, and are there similar concepts?

Chapter 7 . BPMN4SOA specifications are presented as a service oriented process modelling language based on BPMN 2.0 with concepts from SoaML and OOram.

Chapter 8 provides a proof of concept diagram editor realized in Eclipse EMF/GMF for the BPMN4SOA Role Conversation diagram.

Chapter 9 discusses and evaluates BPMN4SOA against the requirements defined in chapter 4.

Chapter 10 concludes the work and provide a discussion on future work for BPMN4SOA and the field.

There are three appendices. Appendix A contains a presentation of elements from OOram. Appendix B contains a selection of meta models from BPMN 2.0 specification that is used in the discussion on BPMN 2.0 and in the mapping between SoaML and BPMN 2.0. Appendix C contains an article written for the course INF5550 at UIO. The article discusses the use of BPMN 2.0 in conjunction with ATHENA, a EU research project.

Research method and scope

This thesis presents a new service oriented process modelling language named BPMN4SOA. The language is based on a discussion and modeling with two new modeling languages from OMG due for release in 2010; SoaML and BPMN 2.0. SoaML is a UML 2 profile, and extends and constrains the use of UML to a service oriented architecture modeling language, and, by means of MDA (See section 3.5.1 on page 18), down to executable PSM. BPMN 2.0 is a process centric modeling language, intended for use in business process modeling and execution. The two languages are known to be a PIM-level language with CIM capabilities and a CIM modeling language with some PIM capabilities accordingly. Parts of BPMN seems very high level CIM, like the Conversation diagram, while other parts of BPMN seems PIM level, like interfaces and operations. SoaML, with its focus on SOA, has CIM level models very closely connected to PIM level and therefore is not that easy to say which level the models are on (PIM or CIM).

2.1 Research method

The research method used in this thesis is based on the third paradigm of computer science research method as described in “Computing as a discipline” by Peter J. Denning (6). The method is called “design” and is rooted in engineering and consists of four steps:

1. state requirements
2. state specifications
3. design and implement the system
4. test the system

The process is expected to be iterated if the system do not meet the requirements.

The method used in this thesis is a variation of this method, and consists for a large part the comparative work on two language specifications and discussion and evaluation on how they perform in their modelling domain. Requirements are created for the modelling languages in question and test them against these requirements. New specifications are then based on the analysis for the design and implementation of the proposed BPMN4SOA.

- Identify requirements
- Create models of the same domain and process
- Identify problem areas, deficiencies and strength
- Evaluate the language against the requirements
- Try and map meta models and concrete graphical models to each other
- Create specifications for a new language
- Implement the proof of concept tool editor
- Evaluate the solution against the requirements

The research is closely connected to the two specifications of SoaML and BPMN 2.0. In addition, the UML 2.1.2 documentation must be used while reading the SoaML specification as reference for the diagrams and meta model. The SoaML specification only describes the extensions and concepts of SoaML.

When working with new language specifications, some issues occur, due to lack of examples and methodologies for how a language should be used to model case descriptions. The practice of modelling, evaluating and remodelling provides insight to the language. This insight is paramount for the stating of the requirements for a modelling language in a particular domain.

One important aspect of the work done in this thesis is the iteration process. To familiarise oneself with two modelling languages not fully supported with tools and few clearly defined modelling methodologies while the specifications of both languages changes in the same process, needs to be an iterative process. Many questions arises and needs to be discussed, tested and evaluated before any new ideas can be brought forth. The ideas for a new language also changes to a great extent during the work.

2.2 Scope

The scope of this thesis is BPMN 2.0 and SoaML, a business process modelling language and a service orientet architecture modelling language. What are their purpose, how can they be used to fulfill their purpose and can they be combined into a new language thus making a more flexible language with one notation.

Table 2.1: Description of values for scope discussion

Value	Scope description
Full	All aspects of abstract and concrete model is in focus
Partial	Some aspects of abstract and concrete model is in focus
None	No aspects of abstract and concrete model is in focus
Abstract	All aspects of meta model is in focus
Concrete	Some aspects of diagram model is in focus

Table 2.2: Diagrams and constructs and the scope for this thesis

Language	Diagram	Scope
BPMN 2.0	Collaboration	Partial
BPMN 2.0	Process	Partial
BPMN 2.0	Choreography	Full
BPMN 2.0	Conversation	Full
BPMN 2.0	Extensions	Abstract
SoaML	Participant Architecture	Full
SoaML	Capability architecture	Full
SoaML	Participant Architecture	Partial
SoaML	Sequence diagram	Concrete
SoaML	Activity diagram	Concrete

Large parts of BPMN 2.0 and SoaML are not discussed in this work. Table 2.1 provides an explanation of the degree of how much a topic is in scope.

The requirement section 4.3 on page 23 lists three main focus areas for the required aspects of a solution; Service model, information model and process model. Service model is the main focus and provides the “who” and “how” of the Zachman framework¹. Service models needs to support some information model capabilities to model the “what” aspect. For modelling the “when” aspect you need a process model. There might be an overlap between the process model and the service model regarding “how” and “who”.

The scope for this thesis is therefore to focus on capabilities that each language brings to the table that supports these three modelling areas: What do they lack in model capability, where do they overlap and how can they be combined and/or extended.

Table 2.2 provide a list of main focus areas for the discussion.

BPMN: For BPMN the main focus is on the high abstraction level model diagrams Conversation, Choreography and Collaboration, with emphasis on the overlapping functionality found against SoaML. The process model diagram is part of the discussion on the process model requirements but any overlap against SoaML (UML) is not a main focus. The scope for the meta model is mainly on discussing particular model elements in some context. How the BPMN

¹Section 4.2 on page 22

specification provides process modelling techniques for BPM, executable processes and BPEL transformation support is out of scope, but this is part of the scoring in the requirements.

SoaML: SoaML extends UML and is therefore capable of modelling all aspects that UML can. Diagrams like state machine, use case and others that is not discussed in the SoaML specification is out of scope. ServicesArchitecture diagram with ServiceContracts, Capability modelling, ParticipantArchitecture with ServiceInterfaces and the information model diagrams like Class (for MessageType stereotype) diagram is in scope for the SoaML discussion. Activity diagram and Sequence diagram is part of the process modelling capabilities of SoaML and is therefore part of the scoring in the requirements.

OOram views and methodology is part of the discussion on the solution and concepts. OOram is considered as the source of the ideas that are given to the BPMN4SOA solution, but OOram will not be mapped or tested against any of the two new languages. OOram provides many views for a domain description, but the role model view, sequence view and the interface view are the main focus.

2.3 Outlining the task

Modelling service architecture and business processes, defining the processes and the scope of the process is paramount for the result. We humans tend to think in models and we choose the appropriate model for a given task subconsciously. We simplify the world so we can better understand it in our daily life. This means that a model cannot be in itself wrong, only good enough, or (not) suitable for the task at hand (25).

The discussion around BPMN 2.0 and SoaML need to define and select a set of business processes that could be represented in both modelling languages. The languages are quite different in a number of areas, but also similar in others. The use of both languages on different processes, identifies the modelling languages' problem areas and their strengths; In what degree are they similar or different? Thus, we need to specify business processes comprising of human to human interaction, human to machine interaction and machine to machine interaction ideally on both the CIM and PIM modelling level. At first glance, BPMN 2.0 seems, having nearly no information model specification abilities, very business process oriented, and SoaML, being a UML profile, seems very technically- and architecture oriented.

There are obvious differences between a modelling language that models business processes where human interaction is the execution environment, and business processes where some computer system is the execution environment. In the perfect world, a modelling language should be executable, so that we can create a model and run it on the fly. But we can not execute a process automatically when humans are involved. Creating a process, for instance at a hospital for taking blood samples, you need to involve a huge staff, have discussions on the topic, or problem areas, to find the best practice and win win situations, formalize the sequence, physically

organize the hospital and train the personnel to perform the business process. In such an application, the modelling tools are more a documentation environment than an execution environment. However, if you have an automated buyer/seller case where an order processor gives instructions to, say, an automated storage collector robot, you can use executable models with more ease. But probably you are more constrained by the capabilities of the machines you control, rather than with humans you can mold and educate to fit a process. Then again, combining humans and machines in the same process can complicate matters even more.

An ongoing discussion regarding the relationship and coexistence of SoaML and BPMN 2.0 is in what areas do they model the same behaviour, in which areas do they differ and how can you transform from one model language to another, i.e. which elements maps to which element. Basically, what are the differences, similarities and overlap of the two modeling languages.

2.4 Tool support

Since BPMN 2.0 is still not finalised, many tool vendors awaits the final specification release before implementing and releasing tool support in their products. Others, like Oracle, have implemented partial support in their JDeveloper². For this reason I had, as of autumn 2009, no working BPMN 2.0 editor available, specifically for the Conversation and Choreography diagrams. I tried to create an editor for these diagrams using Eclipse EMF/GMF for use in the modeling of my cases. The purpose of these diagrams is to have a higher level of abstraction to the business process models, used both for overview and drill down diagram navigation, and in the work process of creating business processes between partners without revealing the inner workings, or private processes, of the interacting participants. However, winter 2010, a company called Signavio inc. released BPMN 2.0 support with Conversation and Choreography diagrams. My work done autumn 2009 creating this diagram editor is then rendered unnecessary for the actual implementation of these diagrams and consequently removed from this thesis work. But by identifying extensions or improvements to BPMN 2.0, I will use the EMF/GMF-editor for implementation purposes.

SoaML has some tool support, however the support is not always according to the last specification. Modelio³ and MagicDraw⁴ are examples of tools. Creating improvements to tools on SoaML is not the scope of this thesis. As of mid April 2010, Modelio supports the latest SoaML specification.

Choosing the appropriate modelling tool is difficult. In my experience, usability is often non correlating with complexity. The time it takes to learn a tool follows the complexity of the tool. Oracle JDeveloper for instance, being a more or less complete developing environment, has a much higher learning threshold than Objectteering, or Modelio in later versions, being a tool used mainly for model creation and Model

²http://blogs.oracle.com/bpm/2009/10/bpm_11g_hands-on-lab.html

³<https://sites.google.com/site/softteams-rd/prototypes>

⁴http://www.magicdraw.com/files/pressreleases/comeosoa/Model_Driven_Solutions_and_No_Magic_Press_Release.htm

Table 2.3: Requirements and support for various modelling tools

Tool	Difficulty	Complexity	BPMN	SoaML	XMI-export
JDeveloper	Hard	Huge	Ltd. 2.0	No	n.a
Modelio	Easy	Modeller	1.x	Yes	UML Yes/BPMN No
Eclipse	Hard	Modeller	1.x	No	BPMN Yes
Signavio	Easy	Modeller	1.x + 2.0	No	BPMN 2.0

Driven Development (MDD). Creating a BPMN model in JDeveloper is not an obvious and easy task to do. The diagrams looks different, compared to the BPMN specification, and the whole diagram creation process is much more complicated than in the tools dedicated to model creation (like Modelio).

Most tools on the market today are commercial. This includes Objecteering, Modelio, MagicDraw and many others. Thus, availability to use a tool for more than a trial period is important for this project.

To be able to create a model and export it in XMI, or some other open format, is important for the model to model transformation process and model to text transformation process; the tools must support some type of export. This is a problem with the first BPMN language specification which do not support XMI export in the standard itself. Thus, the tool vendor has the freedom to create its own model persistence method or export file specification, probably resulting in proprietary solutions. This problem is obvious in for instance Objecteering and Modelio, which has no BPMN diagram export solution. Eclipse Soa bundle, with the BPMN diagram editor from Intalio, uses the open native EMF framework for storage of the model in XMI. There exists a BPMN 2.0 proposal for implementation at Eclipse foundation but the status of this initiative is unclear ⁵.

- All BPMN 2.0 models in this document are created using the Signavio tool
- All SoaML models in this document are created using the Modelio tool
- All BPMN4SOA models in this document are created using the proof of concept tool as described in chapter 8 on page 79

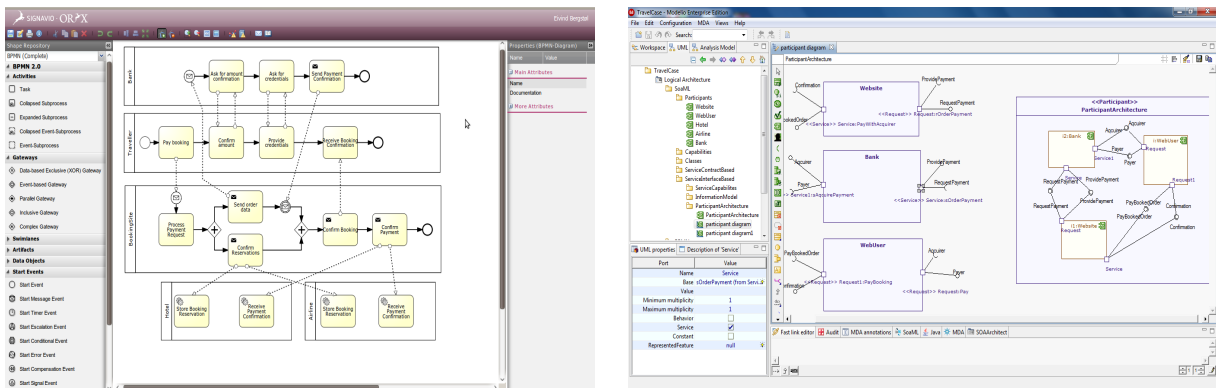


Figure 2.1: Screenshot from Signavio and Modelio.

⁵<http://wiki.eclipse.org/MDT-BPMN2>

Case and domain description

This chapter addresses the case description as a generic explanation of the case example. The Buyer-Seller case is somewhat changed to fit in a travel booking example. The domain of MDD, object oriented and role modelling paradigm and the service oriented architecture (SOA) paradigm are also presented.

3.1 Why this case

The case chosen throughout this document is a version of the Buyer-Seller case. The buyer seller case is chosen because the case is used to exemplify service architecture in many of the model language specifications in use in the service modelling field today. Examples are in the BPMN 2.0 specification (20), SoaML specification (22) and in the SHAPE project (26).

Understandable: Buyer-seller case is easy understood by non IT personnel like business people and administrators. The IT specialist is probably not the best person to decide on business processes and business structures, at least not in the big picture of business administration. But both groups of specialists can relate to the case with ease.

Extensible: Due to its nature it can be very simple, but at the same time be extended to specify a full working case with many different participants, tasks and messages flowing in many directions. In other words, it can be as complex as you need it to be if that is what you want to achieve or as simple as possible just to make a point. This could of course be done with, say, a flowchart of a hospital. But who can relate to that?

3.2 The Travel booking case - outline

A person, or a traveller, accesses a web page, in this case a travel booking page. This portal gives the traveller the ability to book flights, hotels and rental cars all at the same place and same time. This means that the portal plays different roles throughout the process. First as a flight booker, then a hotel booker and finally a car rental booker. The portal does it by communicating with other specialised portals to do the booking. After selecting hotels, flights and cars, the traveller gets a confirmation on the expenses and should proceed to a payment process provided by a payment acquirer, usually a credit card company or a bank.

The following hotel booking procedure is a standard booking process and can be reused on flights and cars rental as well.

Traveller asks for a list of hotel for a given date in a given city.

BookingSite contacts a hotel provider and fetches a list of available hotels and returns the list to the traveller

Traveller looks through the list, picks a hotel and sends a booking request to the BookingSite.

BookingSite then contacts the chosen hotel through the hotel provider and reserves a room.

The payment process follows as described below:

BookingSite provides the traveller with a complete booking order, detailing hotels, flights and car rentals with the order value.

Traveller agrees on the computed price and asks to pay for the booking.

BookingSite informs an acquirer about the order and the amount to pay and who shall receive the money. The traveller is then redirected to the acquirer's web page to pay for the booking.

Acquirer asks the traveller for credit card information and other obligatory information.

Traveller provides all information to the acquirer.

Acquirer processes the card transaction and informs the BookingSite and the traveller about the result. The traveller is then redirected back to the BookingSite

BookingSite provides the traveller with order and payment confirmation documents

BookingSite contacts hotel-, flight- and car rental providers and confirms the bookings.

This case can be created in a huge variety. The descriptive view is just an example and should be considered as such. Some choices made in the process description is done to be able to explore some key abilities in the modelling languages. Human-machine interaction as well as machine-machine interaction, participants playing different roles for different interactions, data specific messaging (hotel and room lists) and many other aspects can be explored in the context of this fairly simple process.

3.3 Business Process Management (BPM)

“A business process is a sequence of activities triggered by a business event such as an invoice, request for proposal, or a request for funds transfer. The process is driven by business rules that involve activities and sub processes. Activities and sub processes are assigned to resources, which are organisational units that are capable and authorized to play specific roles in the process” (4, p. 90).

Business processes are defined by means of some sort of description and the degree of control and flexibility of the process is a direct consequence of this description. Some processes are described using a collection of rules representing the process without specifying any path for the process to follow. Instead, when executed, the process uses events to trigger a path in the fly while a process coordinator is controlling and logging the execution behaviour. This type of systems are very flexible in capturing the dynamic behaviour in business processes. Other processes are described using some sort of graphical modelling language in a flow diagram. The diagram defines all the logic for the given process and execution will follow the specified path. There are three main degrees of autonomy in these kinds of processes: Highly structured, semi-structured and unstructured. Highly structured processes are defined exactly and the path and execution events are all predefined. Semi-structured processes have predefined paths and rules, but the parts and rules may be changed on the fly at execution time. This is also called an ad-hoc process. And last, the unstructured process is a process that have no predefined path of rules and there do not exist any repeatable patterns of rules or sequences among the tasks (4).

There are several business process modeling languages in use today. Integrated Enterprise Modeling (IEM) is developed by Fraunhofer Institute and is used for enterprise modeling and business processes (29). Event-driven process chain (EPC) is a type of flowchart, originally in conjunction with SAP, but is implemented in a wide range of application today. Examples are the Aris Toolset framework and Visio by Microsoft (31). Now, the business process modeling notation (BPMN) is gaining ground in the business process modeling and process execution arena. With the upcoming version 2.0, BPMN is expected to deliver better functionality than the earlier process modeling languages (20).

3.4 The domain of modeling

With the invention of the Internet, new application domains and new types of applications are emerging at a seemingly growing rate. The demand for application developers to quickly and precisely develop new solutions are ever increasing. We create a myriad of mental models to help us understand phenomena of interest and master them. A model is an artifact created for a purpose; it cannot be right or wrong, only more or less useful for its purpose. The choice of phenomena and the questions we ask about them depend on our interests, the modeling paradigms we are comfortable with, and the tools we use for expressing our thoughts. If we want

to precisely communicate our ideas to a colleague, our modeling paradigm and notation must be similar to our colleague's paradigm and notation. If we want to communicate with a computer, our modeling paradigm must be consistent with the applicable computer language. The more expert we are in a particular modeling paradigm, the harder it may be to ask questions and appreciate answers that fall outside that paradigm (25, p. 57).

Some general observations about models:

- A Model is created for a purpose.
- A Model is never complete.
- We tend to think in hierarchical models, even though the world is rarely hierarchical.
- We think in multiple models, always trying to choose the best model for our purpose.

Experts love to discuss whether a certain modeling paradigm is better than another one. Since being an expert means that one has internalized certain ways of thinking, these discussions frequently take the form of religious wars. Reenskaug's view is that it is impossible to evaluate an answer without considering the question, so that a modeling paradigm has to be evaluated on the basis of its effectiveness in a certain context. The best paradigm is the paradigm which best helps one to reach ones goals. If the goals change, one must be willing to reconsider the choice of paradigms. (25)

3.4.1 Objects and the object paradigm

The object oriented paradigm is widespread in the programming community today. It is a proven and effective programming technique used to model structure and relationships in a system. The object is popular due to its nature of being a single memory entity, which gives an object a state. It has properties or attributes, and an object has the ability of encapsulation. All this means that the object can model real life objects with a granularity that we choose to give it. It is this choice that we as programmers have over the object that gives it the power to do what we want it to do in a system. An object is not given by nature but is a simplification that we make of the world for a specific purpose to fit a specific task or a model environment. (25)

By doing abstractions and defining special rules, or interfaces, we can give a set of objects what is called polymorphism. This means that we can use an object without knowing exactly what type of object we are dealing with. For instance we can abstract that all animals has a name and makes a sound. If we create an abstraction of animals that has name as property. We create an interface that has a method called talk. Then we create two different sub classes of animals that implements the interface. When using the talk method in instantiated objects, different sounds should then be returned accordingly. This happens without the need to know exactly what type of object we signal to "talk". (25)

The object can be looked at as having abilities to do things. It can interact with other objects through its methods or procedures, it can change its attributes or properties. However, an object is encapsulated. Meaning that from the outside you can not see what the object does or how it does it, but you can see how you can interact with it. This is the view of an object being a black box and is what gives it the encapsulated property. An example is an object called, say, HotDogMaker. This object has an ability to create and return a hot dog. The method is called `serveHotDog` and returns an object called `HotDog`. How the HotDogMaker creates the HotDog is irrelevant to you as a user of the object; you just need the HotDog as a return value.

An object can not do anything on its own. It has to be invoked in some way. Usually, by another object. This is done by sending a signal from one object to another, a trigger message. The triggered object then performs what ever it is supposed to do, and return a message afterwards. The returning message might be nothing. But as we know, NULL is not void. The HotDogMaker will never just start returning HotDogs without someone asking for it.

Objects has, or can have, states. This means that an object might behave differently from one invocation to another. Say our HotDogMaker object has a timer ticking inside, and this timer stores a value for the property temperature in the HotDog. When invoked `serveHotDog`, then the HotDog served have different temperatures according to the timer inside HotDogMaker. State of the object influences, or might influence, the outcome of interactions with an object.

3.4.2 Role modelling

The object oriented modeling paradigm gives us the tools to model the world in a way that we choose to see it and what fits our need in a particular environment or problem. We then program the application accordingly. The program then instantiates objects from the specified class as defined by the programmer. The class is what gives an object its capabilities in the running program. This way of thinking makes sense for a programmer, but in a model environment we also want to be able to see how objects interact with each other to reach a given goal, the responsibilities and position it has as a part of the whole process. Object oriented modelling falls short in this area. (25)

The role model is a part of a structure of objects which we choose to regard as a whole, separated from the rest of the structure during some period of consideration. A whole that we choose to consider as a collection of roles, each role being characterized by attributes and by actions which may involve itself and other rules (25, p. 71).

A role model is in other words an object oriented abstraction of interactions between instances of classes working together towards a common purpose. It sits on top of our object oriented class hierarchy giving objects a place, a role, in some chosen process.

A role model is the role in a process that a given participant is performing in interaction with other participants. In UML this type of behaviour can be modelled in

the collaboration diagram with collaboration use and roles. The composite structure diagram in UML offer some role capabilities.

“The term “structure” in this clause refers to a composition of interconnected elements, representing run-time instances collaborating over communications links to achieve some common objectives” (19, p. 177).

Based on this way of thinking of objects and roles leads us to the paradigm of service oriented architecture (SOA). Here, we abstract not only memory based objects into participants, but whole systems interacting together in a role model via the Internet. We can, in for instance SoaML, define all classes with capabilities as we choose, but also the interaction and dependencies between them. More of this in chapter 5.2 on page 46.

3.4.3 Services and service oriented architectures (SOA)

Most companies in the world produce some type of service. Even though service providing is not the main business focus for many. A service can be seen as something that an employee does in his or her line of work to some other participant, e.g. a fellow employee or a customer. “A service is any activity or benefit that one party can offer to another which is essentially intangible and does not result in the ownership of anything. Its production may or may not be tied to a physical product” (24, p. 535). This means that the service is not physical or tangible, but something that happens between two or more participants in the instance the interaction is occurring. A service can not be persisted in the sense that it cannot be stocked for later use; A Service is perishable. A specific service can not be separated from the provider; A Service is inseparable. But a service can be formalized in some way, e.g. by modeling the capabilities of the provider for later training of human personnel or new computer systems. When humans provide a service they might vary in quality depending on the person who provides it (24). Hopefully this variability can be solved in computer systems, but the characteristic still applies.

When humans interact with each other, we do this often without thinking about it. But humans can experience problem in a service exchange when we visit other cultures or when language discrepancy occurs. We need to figure out a way to make the communication work and might focus on body language and signaling more than speech (4).

Computers, on the other hand, needs to be directed into how and when to do what in what sequence. To formalize a service we need the idea of an interface, a provider and a consumer. The interface is the invisible barrier that tells the service consumer what type of service and the capabilities of the service that the provider is providing. This establishes a contract between a consumer and a provider and this contract makes the service possible. Without it, a consumer can not know what or how to ask or what to expect in return from the service nor does the provider know how to provide the service to the consumer.

In modern times the idea of a electronic service has emerged. Specifically via the Internet. Examples like airline ticket booking, library cards, toll booths and many more are in use every day. We can divide services into three main groups: Human to human, human to machine and machine to machine (4). Each group can be thought of in terms of the same abstractions, but there are obvious differences.

In service providing there always exists a communication. The consumer knows how to communicate to the provider, and vice versa. If we consider a web service as we know them to day, they are really not a service in that sense; They only respond to a signal, and don't not communicate back. This is a simplified service model comprising of only one interface. This issue is covered in SoaML providing a bidirectional interface, called a ServiceInterface.

3.5 Model Driven Engineering

Wikipedia actually redirects Model Driven Development to Model Driven Engineering. This shows that MDD and MDE might be two sides of the same thing in software development and could be used to discuss the same topic with different names (30). However, engineering is the process of making a practical application of some knowledge of pure sciences¹. Development is the process of analysis, coding and testing of software². The terms are somewhat different and focuses on different stages of the model driven process chain. I choose to understand Model Driven Engineering in this thesis as the process of creating some piece of software tool or model standard that in turn are used in Model Driven Development, regardless of the method used in the Model Driven Engineering process actually is a Model Driven Development process.

When the object oriented paradigm became popular, program development entered a new different era than the statement based programming of the early days. The strive for productivity fueled the use of this technology across the world. In the wake of this, the model driven approach has gained interest as the new paradigm to take over for the object oriented approach. In MDD, a normal work flow would be to analyse the domain of interest, finding objects and roles. The goal is to reduce the design time in each step of the software development process. Then model your software domain in your selected modeling language after which you transform this model into some type of runnable code, your database schema, or just the setup of your classes ready to be implemented. Another important aspect for the MDD work method is separation of concern (2). "It provides the separation of high-level business logic from system's architecture and deployment platform" (17).

In for example Visual Studio .NET, you have a GUI editor for your windows forms. This is a form of model driven development. You create a model, in this case the graphical layout of your form, and Visual Studio generates the code for this form. In early versions of .NET, the code was generated into the class definition it self. However in later versions, this is actually put in a separate file to minimize clutter in the code.

¹<http://dictionary.reference.com/browse/Engineering> Online; accessed 18-Jan-2010

²<http://dictionary.reference.com/browse/Development> Online; accessed 18-Jan-2010

Another example of MDD is the Eclipse EMF where you create a model using a UML modelling tool and import the model. EMF then creates an Ecore model using a model to model transformation script. You can then generate the domain of java classes, ready to be used in your program (7).

3.5.1 Model Driven Architecture (MDA)

The move towards model driven programming that happened around the world was recognised by the OMG as far back as November 2000 (28) when the OMG made public the Model Driven Architecture initiative. The purpose of this initiative was to build a set of standards for MDD domains. The process of MDA is closely related to many other approach to generative programming. Such as the before mentioned Visual Studio generation and Eclipse EMF, but also domain specific languages, software factories, model-integrated computing etc.

“MDA may be defined as the realization of MDD principles around a set of OMG standards like MOF, XMI, OCL, UML, CWM, SPEM, etc. MDD is presently making several promises about the potential benefits that could be reaped from a move from code- centric to model-based practices” (2, p.5).

From this definition, the VS GUI generation mentioned above is not considered model driven architecture because there is no meta model built on MOF for .NET GUI. But the Eclipse EMF like Ecore is built on MOF, and therefor Ecore is part of MDA.

3.5.2 MDD

One key element to MDD, and MDA, is the idea of using transformations to add more detail to the abstract description, refine that description and also convert the description to another representation format. You create a model and transform it, and possibly combine it with other models by means of model transformations. Thinking of code as an expense, the idea of generating the code from models using code templates is one way of reducing the amount of time put into programming code (17).

1. Choose model that corresponds with a problem domain.
2. Subset the model as necessary.
3. Choose models in accordance with the implementation technology platform.
4. Define the interconnection between models.
5. Generate the software system.

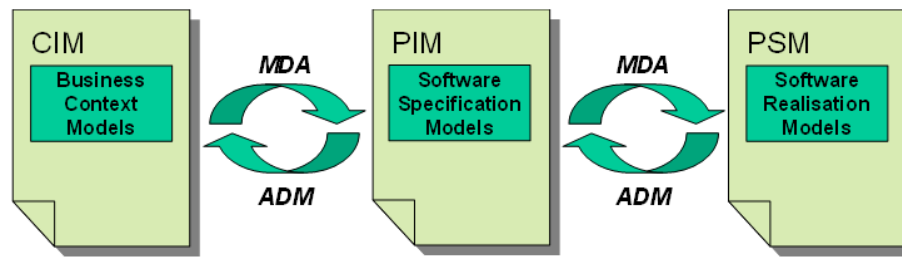


Figure 3.1: Model-driven development (MDD) lifeline
(2, p. 7)

When you need to change the program, you do not change the program code itself, but rather change the domain model and all other models used in the program generation process. If you target a different programming language, for instance, you change only your model for the implementation technology platform and generate your new program accordingly. By using this work flow, the domain model or you implementation model can be reused for different applications, possibly greatly reducing the overall production time (17).

The process of modelling and deployment of a model is normally divided into three abstraction levels: CIM, PIM and PSM.

CIM (Computation independent model) is typically the business context model. BPMN is a typical CIM modelling language. In BPMN 2.0 it can make sense to talk about high level CIM and low level CIM if you think about the differences between a Conversation diagram and a Process model.

PIM (Platform independent model) is the model used for software specification and thus is platform independent. UML is typically a PIM language which has CIM capable diagrams.

PSM (Platform specific model) is the software realisation model. Java, .NET and PHP are examples of programming languages that realise a PIM model into a PSM model.

Requirements for process, information and service modelling

This chapter presents the requirements for a service oriented process modelling language. The three sets of requirements provided here for use in the language evaluation are based on what the BPMN 2.0 and SoaML provides in terms of modelling capabilities and on the position they assert in the Zachman framework (33) (32).

4.1 Descriptions from specification

SoaML and BPMN 2.0 are both modelling languages designed for the Service Oriented paradigm. They have several differences in both the graphical notation and the model driven development work flow. BPMN 2.0 is a process oriented language, while SoaML is a service oriented language with diagrams supporting the process architecture design phase. Both languages model at different abstraction levels and support a wide range of granularity.

4.1.1 Short description SoaML

As discussed in the SoaML section 5.2 on page 46, SoaML supports both an IT and a business perspective on SOA. There is a variety of technologies and approaches in the field of SOA today, many of which embraces a system view of architecture. Systems can be organisations, community processes and information systems. All these systems can be service oriented and modelled through an architecture model separating the concerns of what needs to get done from how it gets done, where it gets

done or who or what does it. SoaML embraces the low level technical implementation of services through a range of UML constructs and diagrams and can be realised through various MDA methods. In addition, SoaML provides extensions to UML diagrams to understand and model a community, process or enterprise as a set of services related to each other supporting the service oriented enterprise with service-enabled systems. Through the support of low level service modeling and supporting MDA technologies joined with the business architecture diagrams, SoaML leverages the mapping of business and systems architecture separating the concern of the architecture from the implementation and technology (22).

4.1.2 Short description BPMN 2.0

BPMN is as discussed in section 5.1 on page 29 first a process oriented modelling language created for business process modeling. The notation provided is created using graphical elements and logic readily understandable by all business users, from the business analysts to the technical developers that implement systems running the process and then for the business people responsible for monitoring and management of the process. BPMN creates a bridge for the gap that exists between the technical process implementation and the business process design. Through support for XML execution languages, such as WS-BPEL (Web Service Business Process Execution Language), BPMN provides the support for process modeling and execution. This is done mainly through process modeling in Collaboration diagrams and Process diagrams. BPMN supports two diagrams for higher abstraction level business architecture modeling through Choreography and Conversation diagrams. The Conversation diagram is used as a bird's eye view of the whole process chain, and supports a limited set of graphical elements. The Choreography diagram provides the ability to specify protocols for interaction between participants of the process, while the Collaboration and Process diagram specifies the actual process taking place (20).

4.2 Zachman Framework

The Zachman framework is an Enterprise Architecture framework that classifies all organisational aspects into a highly structured matrix. This matrix consists of six columns with communication questions (What, How, When, Who, Where, and Why) and six rows containing all the transformation stages from an abstract idea to an instantiation. The intersections in the matrix then contains the classification, i.e. the description of something (17) (33).

Different OMG standards cover different parts of the framework many of which overlap a great deal. Figure 4.1 on the facing page¹ shows the layout of the framework and the taxonomy. The rounded rectangles inside depicts how UML, SoaML and BPMN 2.0 cover different areas of the Zachman framework (5). The SoaML blob in

¹Based on image from: http://en.wikipedia.org/wiki/File:Zachman_Framework_Detailed.jpg Accessed:16 April 2010. Coverage for UML, SoaML and BPMN 2.0 is based on Figure 1 as stated in (5)

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>
Objective/Scope (contextual) <i>Role: Planner</i>	List of things important in the business	List of Business Processes	List of Business Locations	List of important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) <i>Role: Owner</i>	Conceptual Data/ Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) <i>Role: Designer</i>	Logical Data Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure BPMN 2.0	Business Rule Model
Technology Model (physical) <i>Role: Builder</i>	Physical Data/Class Model	Technology Design Model SoaML	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Representation (out of context) <i>Role: Programmer</i>	Data Definition	Program	Network Architecture UML	Security Architecture	Timing Definition	Rule Speculation
Functioning Enterprise <i>Role: User</i>	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

Figure 4.1: View of the Zachman Framework

the figure could be covering parts of the “who” column if we look at the column in a role sense rather than a people sense in both the conceptual and the logical row since ServiceContracts and Architectures with collaboration diagram model parts of these concepts. We also see from the figure that BPMN 2.0 has no data specific constructs, but it could be argued that through the element Message and DataObject that BPMN 2.0 can model the distinction between a physical object and a message with some meaning, but cannot specify it in any more detail.

4.3 Requirements for a service oriented process modelling language

The definitions of the requirements for a modelling language that can model both a service oriented architecture and business process are based on the strengths of both SoaML and BPMN 2.0. They both provide abilities in both SOA and BPM but with different notation and detail degree.

The requirements should cover:

Service modelling strives to create models for the analysis, design and architecture of software in an organisation or cross organisational architectures. The notation should cover both the business domain and the technical domain and the abstraction of software into entities that play some role in the service collaboration. In service modelling, the language should focus on “who” do the participants communicate and on “what” topic do they communicate. The use of composite structures to reuse other service entities to build a new service entities should also be supported. A service is the exchange of value between someone. This means that the service model relies heavily on the information model and the process model to provide the sequence and the data of the messages.

Table 4.1: Requirements for service modelling

Service modelling requirements
Role and Entity distinction
Reusable service model elements
Service interaction protocol
Service provider/consumer
Initiator of a service
Interface definition
Composite structures

Information modelling is the representation of concepts, relationships, constraints, rules and operations to specify data semantics in a system ¹. The information model describes the actual data (or message), structures, attributes it has and the operations it can perform and how the classification of this object relate to other classes.

Process modelling is the sequence of tasks or activities and interactions that are being performed and choices made in the process. The process provide the “how” a service entity is used in conjunction with other entities according to the service model. Additionally, the process model provides the internal processes for the entities. A Process model should consist of a participant, activities, messages, and gateways and the model should provide the process direction and sequence.

4.3.1 Requirements for service modelling

Table 4.1 provide the chosen requirements for this discussion.

Role and Entity distinction is the ability for the language to provide a strongly typed distinction between the concept of a Role and Entity. An entity in this sense is a specific participant, e.g. specific person or specific enterprise by name. The role is then something that the entity is doing in a particular interaction.

Reusable service model elements is the ability of the language to abstract specific contracts or processes into a generic interaction usually in some role model. This could be done either by some pattern in the meta model or to be defined as rules in the specification for a tool implementation.

Service interaction protocol is the specification of a sequence that a particular service must follow to provide the service. This protocol can be viewed as a part of the process model, but is a sub set defined only for a separate concern.

Service provider/consumer defines who is the user of a service and who is providing the service to the user.

¹http://en.wikipedia.org/wiki/Information_model

Initiator of a service is the one that starts, or initiates, the service communication. This is usually the consumer of a service, but one can imagine examples where the service provider is the initiator of the service.

Interface definition is the definition of what a service provider or consumer can perform in some interaction (The “invisible” layer that exists between the parts of a service). The interface is typically made up of operations, and thus is connected to the information model part of the language. Operations provided by the interface is typically part of the interaction protocol.

Composite structures provides the language with the ability to define internal service architectures or processes for roles or entities.

4.3.2 Requirements for information modelling

Table 4.2 provides a list of definitions that should be covered in an information modelling language: Concepts, relationships, attributes, operations and constraints.

Table 4.2: Requirements for information modelling

Information modelling requirements
Definition of concepts
Definition of relationships
Definition of attributes
Definition of operations
Definition of constraints

The concept is typically a classification, or an abstraction that types some object. There exists an instance-of relationship between class and object. There exists a part-of relationship with the aggregation abstraction. An aggregate class is typically a class that has many attributes defined by some other class. Generalization is an abstraction typically implemented by inheritance. Similar objects uses an is-a relationship to establish common ancestry thus a constituent object is a specialisation of the generic object. The last relationship form is the member-of relationship, or association. This is the abstraction where a set of objects are considered a higher object (16).

The attribute or property of a class is a type of aggregation or association relationship, but are used as something intrinsic to an object. Attributes can be thought of as something that does not have any existence outside of the object, e.g. a name.

Operation is a dynamic property of an object typically able to perform some task on the object itself or with other objects.

Rules or constraints constitutes binding laws on objects and/or sets of objects, e.g. min/max value for properties, cardinality of relationships or ordering of lists (12).

4.3.3 Requirements for process modelling

Table 4.3 on the next page provides the chosen requirements for this discussion.

Table 4.3: Requirements for process modelling

Process modelling requirements
Participants and activities/tasks
Decision gateways
Event and error handling
Support for automated/human tasks
Model transformation and execution
Multi layer modelling
Public and private processes

Participant and activities/tasks represent the modeling capability of “who” does “what” “when”. Participants contains tasks being performed in some sequence.

Decision gateways are if-elseif-else constructions for logical choices and thus provide several possible paths that a process can be executed through, depending on decisions based on some information data.

Event and error handling is the ability to model events and resolve exceptions in a process.

Support for automated/human tasks defines a strongly typed distinction between computer performed tasks and human performed tasks.

Model transformation and execution means that the language can be used for model driven engineering and the existence of a Meta Object Facility (MOF) based meta model for the language so that the language can exist in a MDA.

Multi layer modelling is the ability for a process language to provide the same process at different levels of granularity and the means to couple the processes together in a bidirectional manner. This provide the means for a cross-business process, public process or view process definition as discussed in appendix C.2 on page 114.

Public and private process is a sub requirement to multi layer modelling, but differ in the sense that it only require two levels. Multi layer modelling leans more against different diagram types being used, while public and private processes can be made within one diagram type by using a modelling technique.

4.4 Scoring of requirements

The evaluation of BPMN, SoaML and BPMN4SOA will produce scores for each requirement. This scoring is based on an evaluation of the language specification and case model and a short discussion on the requirement. The scoring is done from 1 to 5 and is as follows:

1. There exists little to none functionality to cover the requirement.
2. The topic is covered in some extent but seems too simple or incoherent.
3. The language do not excel nor ignore the requirement.
4. There exists good capabilities covering this requirement, but there are shortcomings.
5. The language focuses greatly on this requirement and is very good to fully compliant.

Each language evaluation summarize scores for each modelling area (Service, information and process) and also creates a total score for the evaluated language.

Evaluation of BPMN 2.0 and SoaML

This chapter presents and discusses parts of the specifications of both BPMN 2.0 and SoaML. The evaluations of both languages have three parts: First, a presentation of the language, in terms of model constructs and/or selected parts of the meta model. Second, a discussion of the model realization for the case, as described in chapter 3 on page 11. And finally, the requirement evaluation of the language. The chapter is concluded with a requirement evaluation of both languages side by side.

5.1 BPMN 2.0

Business people are comfortable with flow-chart based business process visualizations today. The degree of details in the charts differ, but often they are quite simple. On the other side, creating an executable process needs a higher level of detail to be able to execute. The need to bridge this gap is evident and BPMN provides the formal mechanisms to map the appropriate visualisation of a business process to the appropriate execution platform (20).

Business Process Modeling Notation 2.0 (BPMN 2.0) is based on numerous modeling languages like UML Activity Diagram, UML EDOC Business Process, IDEF, LOVeM, RosettaNet, Event-Process Chains (EPCs) and Activity-Decision Flow. Object Management Group have reviewed these existing notations and extracted the strengths from them to create the BPMN 2.0 standard. The focus for BPMN 2.0 is to provide a notation that is easy to understand for both business process personnel and other participants using or participating in the process as well as giving support for process execution in a XML language e.g. WSBPEL¹ (20).

¹Web Services Business Process Execution Language

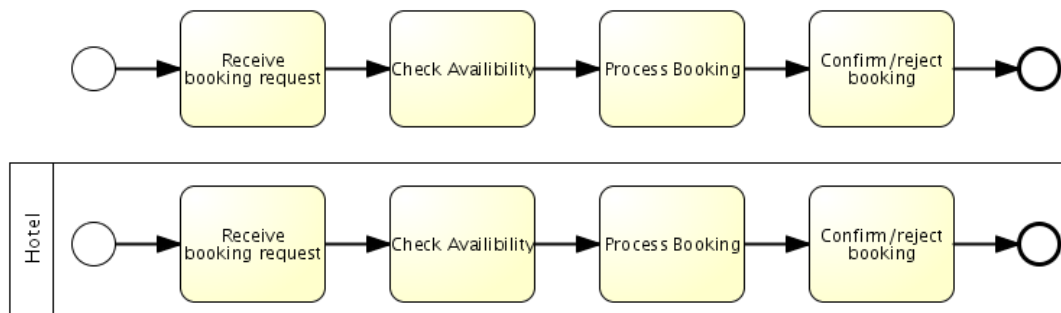


Figure 5.1: Example of a simple private process based on a travel case created for documentation purposes. One is created outside a participant, the other inside the participant (Pool).

5.1.1 Usage

BPMN provides three diagram types: Processes, also known as orchestrations, Choreographies and Collaborations. Orchestration processes include the executable and non-executable private processes as well as the public process. Collaborations may include processes and/or Choreographies. A special type of Collaboration diagram is called a Conversation diagram (20). This division can be confusing, because the difference between a process and collaboration is not very visible. I therefore discuss the three diagram types of Collaboration, Choreography and Conversation.

The BPMN specification does not give strict constraints on how a modeler wants to model his/her process or on what level of detail or granularity should be chosen. A participant may function just as well as an organization as some type of low level computer component. A typical process development methodology could be to specify the general outline of the process as activities and participants and then add more detail to the model to prepare the process for execution (27). BPMN is a CIM language, but it has the ability to detail web services with operations thus has some, but limited PIM capabilities. This ability is only possible in the meta model and has no defined diagram notation. Mapping from BPMN 2.0 models to WS-BPEL is formalised in the specification (20, ch. 15), but the mapping is out of scope for this discussion.

A private process, or work flow process, is a process that is internal to a specific organization. If the process is executable, the process is modeled according to the needs for the execution to work and is more detailed than a simple documentation process.² A process does not need to have participants specified as pools. But if pools are present, the process needs to be modeled inside the pool to be a private process.

A public process is modeled using message flows to and/or from another participant, thus providing documentation for externally available interactions. A public process is modeled with messages flowing between the private process to some other participant. Only the tasks that actually exchange messages are included in the

²See figure 5.1 for example

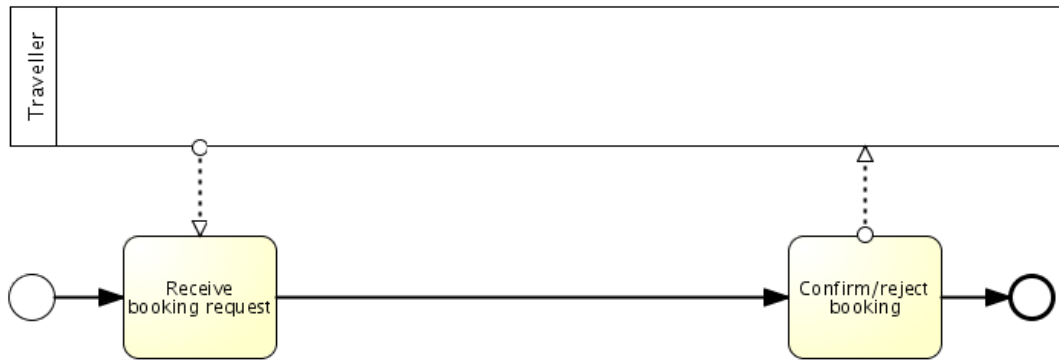


Figure 5.2: A public process showing the MessageFlows that exists between the traveller and the process.

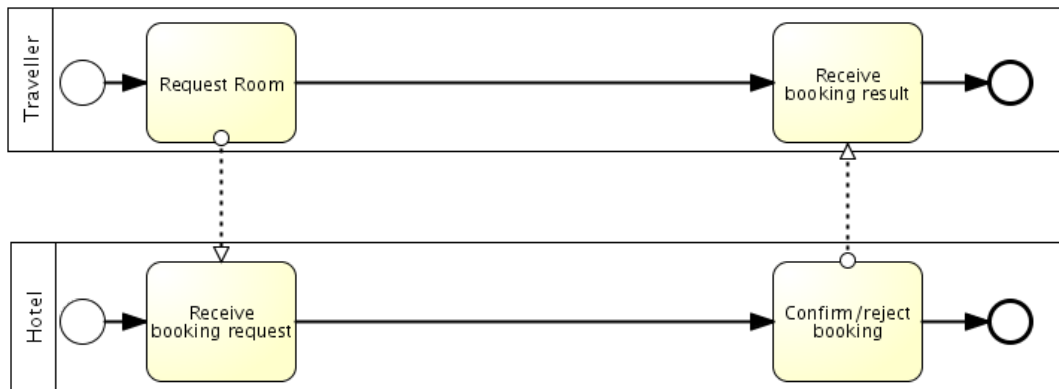


Figure 5.3: A public process showing the message flows that exists between the traveller and the process.

public process model. Thus, showing only the public process hiding the private inner workings. As seen in figure 5.2, I have purposely left a space where the two private tasks of the hotel was located in figure 5.1 on the facing page, showing only the message exchange occurring in the process between the Traveler participant and the process.

If we have two or more participants exchanging messages we have a Collaboration. The public process depicted in figure 5.2 is a type of Collaboration, but ideally this process should include the name of the participant containing the actual process. Figure 5.3 includes the tasks that the traveller goes through to exchange the messages to the hotel. All tasks are public, and as we know from the private process, the hotel executes more internal tasks than shown in the public collaboration process.

BPMN 2.0 provides an extra diagram type called a Choreography. A Choreography is a higher level of abstraction depicting presence of message exchange between two or more participants. A Choreography has no pools or MessageFlow notation elements, and BPMN provides a special type of graphical notation for the Choreography. It is intended to show the expected behavior of a process, or protocols for interactions. One ChoreographyTask differs from the normal message flow ex-

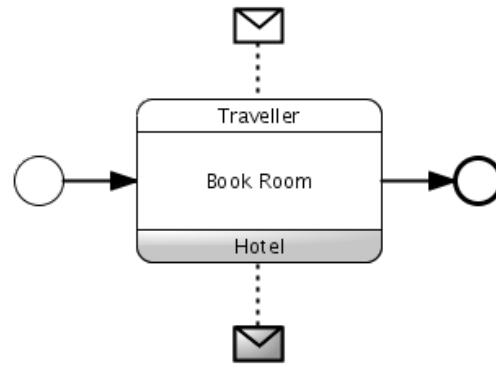


Figure 5.4: A simple Choreography between a traveller and a hotel initiated by the traveller

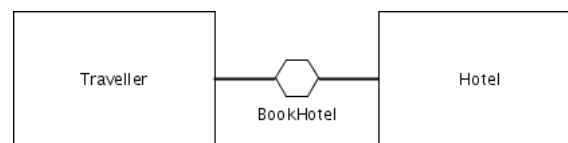


Figure 5.5: A simple Conversation between a traveller and a hotel

change we see in Collaboration because a ChoreographyTask can contain a set of MessageFlows, grouping them together. It also contains the information of who is starting, or initiating, the process. As shown in figure 5.4 the traveller is initiating the room booking process with the hotel. We do not, from the notation itself, see the actual tasks executed in neither participants and we don't see the actual messages exchanges. But we know from processes above that there is some more to the diagram than is shown. The actual messages exchanged is depicted as envelopes attached to the participant coloured accordingly. A ChoreographyTask can be sequenced with other ChoreographyTask using the SequenceFlow arrow as in orchestration processes. This is the simplest form of a Choreography. Reference appendix figure C.5 on page 117 for a more comprehensive example of a Choreography.

Finally, BPMN 2.0 provides an extra diagram type called a Conversation diagram which is a limited Collaboration. The Conversation diagram shows only a bird's eye view of the process. Participants exchanging messages show these messages as a hexagon, depicting that there are messaging taking place. If messages are related to each other, several hexagons can be shown between the participants. In the example of the traveller and the hotel, the conversations regarding the booking should not be in the same dialog hexagon as the conversation happening when the traveller arrives as the hotel to check in. Figure 5.5 shows the simplest form of conversation diagram for the message exchanging in the simple example used in the presentation of BPMN diagram types. For a more comprehensive example, reference appendix figure C.4 on page 117.

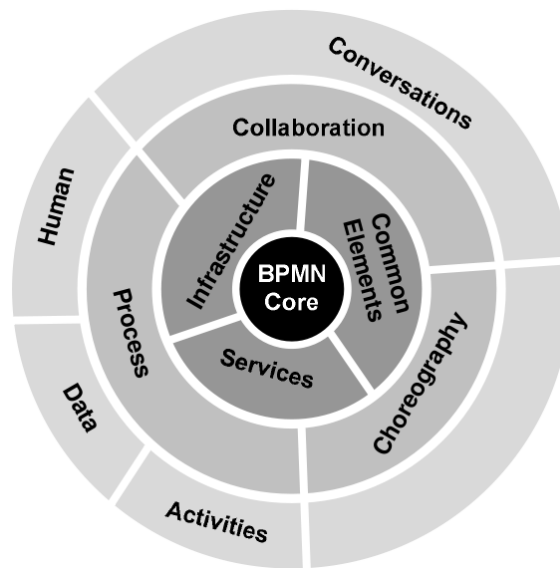


Figure 5.6: A representation of BPMN Core and Layer Structure

5.1.2 Constructions and their meta model

This section do not include a complete list of diagram elements that exists in BPMN 2.0, but only discuss elements that are linked to the requirements to modelling languages as stated in 4.3 on page 23.

“The proposed technical structuring of BPMN is based on the concept of extensibility layers on top of a series of simple elements identified as *Core Elements* of the specification” (20, p. 39). On top of the core, extensions are made to create the basic building blocks for all diagram elements, concepts or constructs. Figure 5.6 shows the basic principles of the layering from the core to the diagram types in the specification. One result of this layering is that extensions can be made on top of one or more layers while still be compliant to the BPMN 2.0 conformance for a given tool thus different tool vendors can build layers for specific domains on top of the BPMN 2.0 specification.

5.1.2.1 Extensibility

Extensibility is introduced in BPMN 2.0 as a new feature for extending diagrams with new notation types or extra meta model elements used for various purposes (See figure B.3 on page 104). It can be used by tool vendors to make some specific functionality. The idea of the extensions is that the extensions can be made without making changes to the BPMN meta model itself, preserving the conformity to BPMN and thus the interoperability of diagram across tool platforms. The extensions will then be unique to the tools that the extensions is created for, while all processes or models will still be according to BPMN specification. Extension to the graphical notation should preserve the look and feel of BPMN so that other modelers or viewers not familiar with the extensions intuitively understand the diagram (20).

A BPMN Extension consists of four different elements:

- Extension
- ExtensionDefinition
- ExtensionAttributeDefinition
- ExtensionAttributeValue

“The core elements of an Extension are the ExtensionDefinition and ExtensionAttributeDefinition. The latter defines a list of attributes which can be attached to any BPMN element. The attribute list defines the name and type of the new attribute. This allows BPMN adopters to integrate any meta model into the BPMN meta model and reuse already existing model elements” (20, p. 47).

5.1.2.2 External Relationships

BPMN is intended to provide elements required for the “construction of semantically rich and syntactically valid Process models to be used in the description of Processes, Choreographies and business operations in multiple levels of abstraction” (20, p. 82).

Processes exists in many different environments or complex business systems. The External Relationships elements intention is to give BPMN 2.0 the ability to integrate other systems elements in a non intrusive manner (See figure B.2 on page 104). This means that other elements can be related to each other by linking. An example of this is a UML use case relates to a particular process, or a UML class model relates to a particular BPMN message.

5.1.2.3 Conversation Assosiation

A ConversationAssosiation is used in Collaboration and Choreography diagrams to map message flows to a Conversation thus grouping the message exchanges occuring in the diagram (See appendix figure B.1 on page 103). As discussed above, message flows that are occurring between two participants do not always represent the same conversation. In a Collaboration or a Choreography, the modelling tool should give the modeler the ability to group message flows by means of ConversationAssosiation by utilizing the CorrelationKey construct.

5.1.2.4 Interaction Specification

“The interaction specification element is the superclass for all elements that require definitions about interactions between Participants” (20, p. 80). This include the Collaboration diagram, Choreography diagram and the Conversation diagram. It contains the list of participants and the list of message flows in Choreography, Conversation and Collaboration diagrams (See appendix figure B.4 on page 105). One interesting element to note in this meta model is the choreographyRef association from Collaboration to Choreography and the conversationRefs from Choreography and Collaboration to Conversation which binds the diagrams together.

5.1.2.5 MessageFlow

The MessageFlow diagram element shows the exchange of a Message between to Participants (See appendix figure B.5 on page 105). It can only connect to a participant represented as pools or to activity tasks that do not reside in the same process. A MessageFlow can consequently not exist between two tasks inside the same pool or between two tasks in a public/private process. In other words, it can only exist between boundaries of pool or must cross at least one pool boundary.

The messageflowref association between a ChoreographyTask and the MessageFlow provide the constraint that one MessageFlow can only exist in a single ChoreographyTask and it exist a message in a ChoreographyTask.

5.1.2.6 Participant

The participant represents some type of organisation or role (See appendix figure B.6 on page 106). The participants notation differs from diagram to diagram. In a Collaboration it is represented by the pool, able to contain a process. In a Choreography, the participant is a part of the ChoreographyTask and in the Conversation diagram it is represented by a simple box and not able to contain any process.

An aspect of the participant is the PartnerEntity and PartnerRole concepts. Specifying a role or entity on a participant can provide additional or different abstractions to different diagrams. A Participant can only play one role at a time or be defined as one entity. If Role or Entity is given, the name of the Participant should be changed accordingly.

The participant also references the concept of Interface, thus providing a set of communication access points that MessageFlows can, by means of Messages, use to give concrete models web service executability.

Another important concept is the ParticipantAssosiation. In different diagram participants can be modelled differently while representing the same entity. The specification of BPMN 2.0 gives three cases where this is applicable (20, p. 95):

- A Collaboration references a Choreography for inclusion between the Collaboration's Pools (Participants). The Participants of the Choreography (the inner diagram) need to be mapped to the Participants of the Collaboration (the outer diagram).
- A Collaboration references a Process (within one of its Pools) and that Process contains a Call Activity that references another Process that has a definitional Collaboration. The Participants of the definitional Collaboration for the called Process (the inner diagram) need to be mapped to the Participants of the Collaboration (the outer diagram). Note that the outer Collaboration may be a definitional Collaboration for the referenced Process.
- A Choreography contains a Call Choreography Activity that references another Choreography. The Participants of the called Choreography (the inner

diagram) need to be mapped to the Participants of the calling Choreography (the outer diagram).

5.1.2.7 Services

The meta model for Services enable the modeling of services as interfaces with operations and in and out messages (See appendix figure B.7 on page 106). The Interface can be referenced as Participants or CallableElement.

5.1.2.8 Tasks

“A Task is an atomic Activity within a Process flow. A Task is used when the work in the Process cannot be broken down to a finer level of detail. Generally, an end-user and/or applications are used to perform the Task when it is executed” (20, p. 133). There are several types of tasks, as we can see from the meta model: SendTask, ReceiveTask, ServiceTask, UserTask, ManualTask, ScriptTask and BusinessRuleTask (See appendix figure B.8 on page 107). A task not specified as of any type is called an AbstractTask and is mainly used in descriptive BPMN modeling. (The focus will only be on SendTask, ReceiveTask and ServiceTask meta models in this discussion.)

5.1.2.9 ServiceTask

The ServiceTask is special in the sense that it has a specific operation from an Interface as a property (See appendix figure B.9 on page 107). As we learned above, the Interface is an attribute of a Participant or a CallableElement, meaning that the pool containing a ServiceTask must have reference to an Interface.

5.1.2.10 SendTask and ReceiveTask

The SendTask and ReceiveTask can be viewed as the same as a single service task divided into two separate tasks (See appendix figure B.10 on page 108). This adds more process functionality in the sense that you can receive on one interface’s operation, do more processing and then send messages back at a later stage. A ReceiveTask waits for a message to arrive, and in the case where no start event is present, instantiate a process. Both send- and receive tasks finish as when a message is sent or received accordingly.

5.1.2.11 Conversation diagram

The Conversation diagram is similar to the Collaboration diagram but participants cannot contain any processes, and MessageFlows are grouped together into a hexagon element depicting the existence of a conversation between the participants. The messages are related to each other and “reflect distinct business scenarios” (20,

p. 329) often as request-response or can be long running complex business communications.

The Conversation diagram meta model show us the InteractionSpecification element containing the Participants and the MessageFlow element while the ConversationContainer holds the ConversationNodes (See appendix figure B.11 on page 108). The ConversationNode reference Participants through participantRef.

5.1.2.12 Choreography diagram

The Choreography diagram differs from a process orchestration in the sense that it provides the protocol for cross business communication. While the Conversation diagram says that there exists a conversation, the choreography provides what and in what sequence the interaction is being performed. “A Choreography is a definition of expected behavior, basically a procedural business contract, between interacting Participants” (20, p. 342).

The InteractionSpecification provides us with the Participants and the MessageFlows (See appendix figure B.12 on page 109). The ChoreographyTask is a FlowNode just as normal process tasks are and thus is provided through the FlowNodeContainer. The ConversationAssosiation is also evident in the meta model providing the connection from choreography to conversation.

There must exist at least two Participants on a ChoreographyTask and one of which is the initiating part.

5.1.3 Travel case - BPMN 2.0

The modelling process of this case has been an iterative process. First models consisted of many different event types and too low granularity level. The case models are the results of creating a BPMN 2.0 model, then a SoaML model, when evaluating them against each other to refine models in both languages.

The analysis of a possible process starts by identifying the participants and what they are communicating about and finally creating a Conversation diagram based on that analysis. This is the top-down approach for modeling architecture based environments. BPMN is a business process language, meaning that the focus is on “how”, and “who” does it “when”, rather than “what”. For instance, in specifying a computer protocol, BPMN can provide the sequence for the protocol but not any data specific models.

Figure 5.7 on the next page presents five participants: The traveller, the booking site, a bank, an airline and a hotel. Mainly the traveller uses the booking site to order both a hotel and a flight but all participants communicate with a bank for the money reservation and transfer. For simplicity all payment go through one communication, but this is potentially a huge process model.

The booking site provides the means for the traveller to book the hotel and the flight for the trip and the actual communications to the airline and the hotel is done by

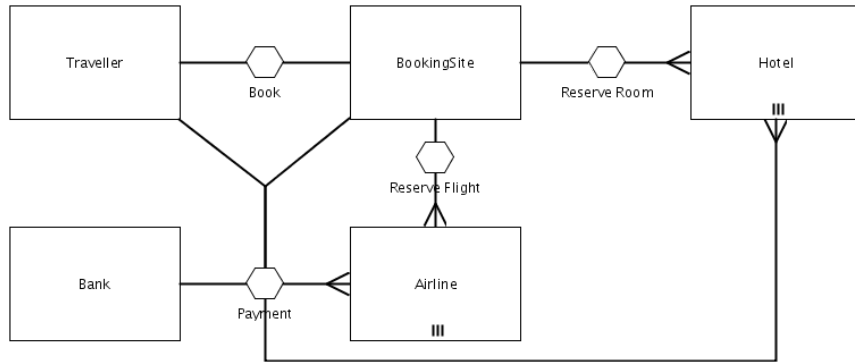


Figure 5.7: Travel case Conversation

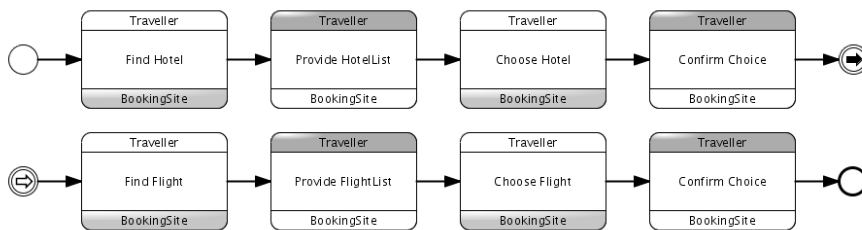


Figure 5.8: Choreography between traveller and BookingSite to book hotel and flight

the booking site. Typically, the booking site provides an order object to the traveller to pay. This payment is then put through a payment service provided by the bank and a message from the bank is sent to the booking site. The airline and the hotel should be provided with a confirmation on the payment status and act accordingly. When the traveller travel with the airline and arrives at the hotel, both should be able to know the status of the bank transfer. Thus, the they should be included in the payment communication.

Note that the conversation diagram displays the participants hotel and airline as multiple participants. This makes sense because there are several hotels and airlines providing the same kind of service.

With all the participants identified, we create the public processes that exists between the participants. The public processes will be divided into several smaller message exchanges. At this level the choreography diagram in BPMN 2.0 has the ability to separate the conversations into smaller blocks (a choreography process) detailing specific parts of the whole. First we look at the choreography between the traveller and the booking site.

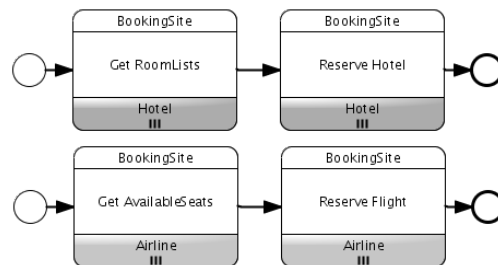


Figure 5.9: Choreography between booking site and hotel/airline.

When modeling a choreography it is probably a good idea to use the “Verb-Noun” naming convention for the tasks being performed (27) .

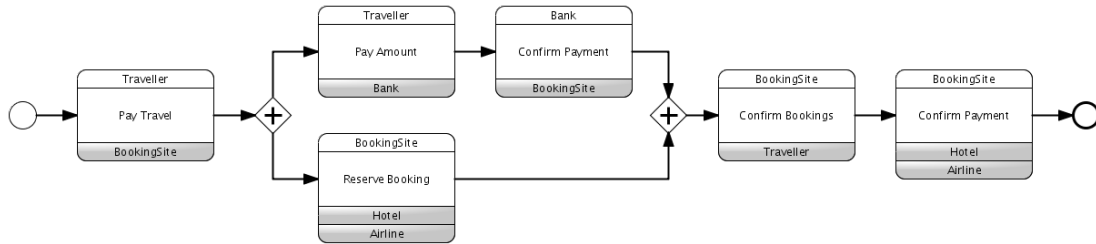


Figure 5.10: A Choreography between all participants detailing the payment process

Figure 5.8 on the facing page provides us with the initial abstract interaction between the traveller, booking a hotel and a flight, and the booking site. It provides messages for getting a list of hotels and flights the traveller can choose from and the acknowledgement of the choices made by the traveller. The intermediate throw and catch link events is just to make the model more organised. The participants with the white background is the initiating part of the message exchange in each ChoreographyTask.

One ChoreographyTask can contain messages going both ways within the task. This means that the return messages, like flight list and hotel list, might be modelled in one task. By modeling that way, we could remove all the tasks where the booking site is the initial participant. How this should be done is not clearly defined in the BPMN 2.0 specification and is probably a choice to be made by the modeller (20). Here, the chosen technique is modelling with several tasks, because, as we have seen in the conversation diagram above, the booking site contacts several hotels and airlines to provide the list to the traveller, e.g. room availability checks. Figure 5.9 on the preceding page provides this interaction. This interaction is considered simpler, e.g. a RPC operation to a web service. Thus, they are modeled with just one task each.

The choreography in figure 5.10 display the payment process. The traveller initiates the process by asking for payment routines. Typically a web site will redirect the user to a payment process and while the customer is in the payment process the site reserves the bookings with the hotel and/or airline. Later on, the confirmation of the payment from the bank arrives at the booking site or to the hotel and airline. In this case the booking site informs the hotel and/or airline about the payment. They in turn has some private process to deal with an incoming payment and register that payment to a particular booking.

Figure 5.11 on the next page displays the process of booking both hotel and flight. There are three choices made here for the process interaction. The traveller is typically a person and all tasks performed by the traveller are normal tasks with no type specified. The messages going from these tasks to the booking site is typically a Website navigation. Therefore, the booking site is modeled using the message events. When messages is sent by the traveller to the booking site, the site waits for a message and resumes process execution at that point. When the booking site contacts either the hotel or the airline to get list and confirm booking, the communication is based on a simple web service communication with request-response

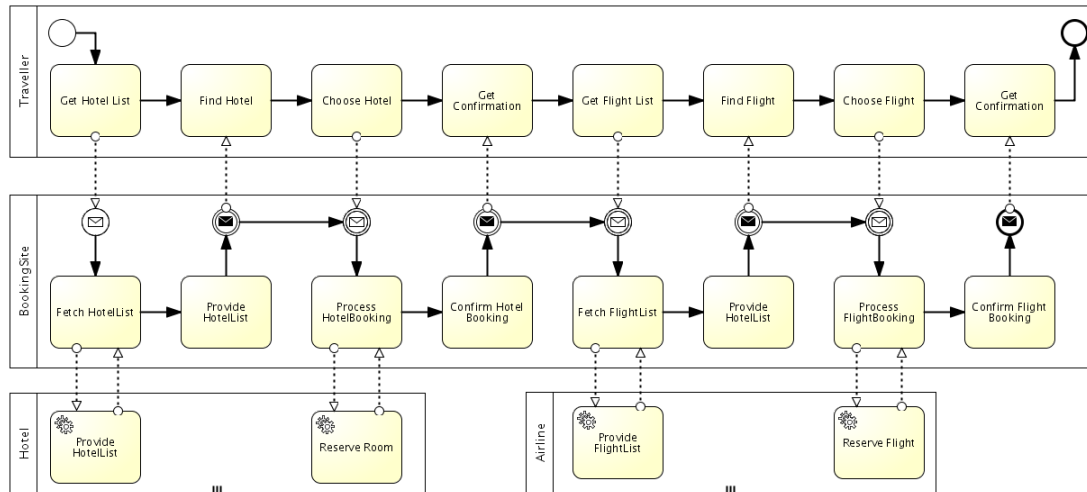


Figure 5.11: A BPMN 2.0 Process detailing the choreography in figure 5.8 on page 38 and 5.9 on page 38

between two tasks at different participants. Note that when a service task is used as in the hotel pool, there is no need for start- and end events. The process created here is according to the descriptive level of modeling defined by Silver (27).

Figure 5.12 on the facing page display the process where the traveller will pay for the booking of the hotel and the flight. In the choreography of the payment process, we can identify the message flows between participants in both orchestration and choreography processes as representing the same message exchange. We also see the same parallel gateway in the booking site participant as in the choreography. In the payment process, the send and receive task are being used but represent the same pattern as one send or receive event and one task performing some processing. In the communication between the booking site and the hotel and airline there are no return message. In an execution environment this would probably be void or a boolean return. Compared to the get available rooms list task in the booking process, a void or a boolean return value should not be modeled as a message as opposed to a list of data.

5.1.4 Requirement evaluation

There are three focus areas of the requirements for a modelling language outlined in section 4.3 on page 23: Service, Information and Process modelling.

5.1.4.1 Service requirement evaluation

Service modelling is not the main focus for BPMN 2.0, but there are many constructs that can be thought of as service providing. BPMN 2.0 do have the concept of a web service. BPMN 2.0 is expected to do quite well in these requirements, but not excel.

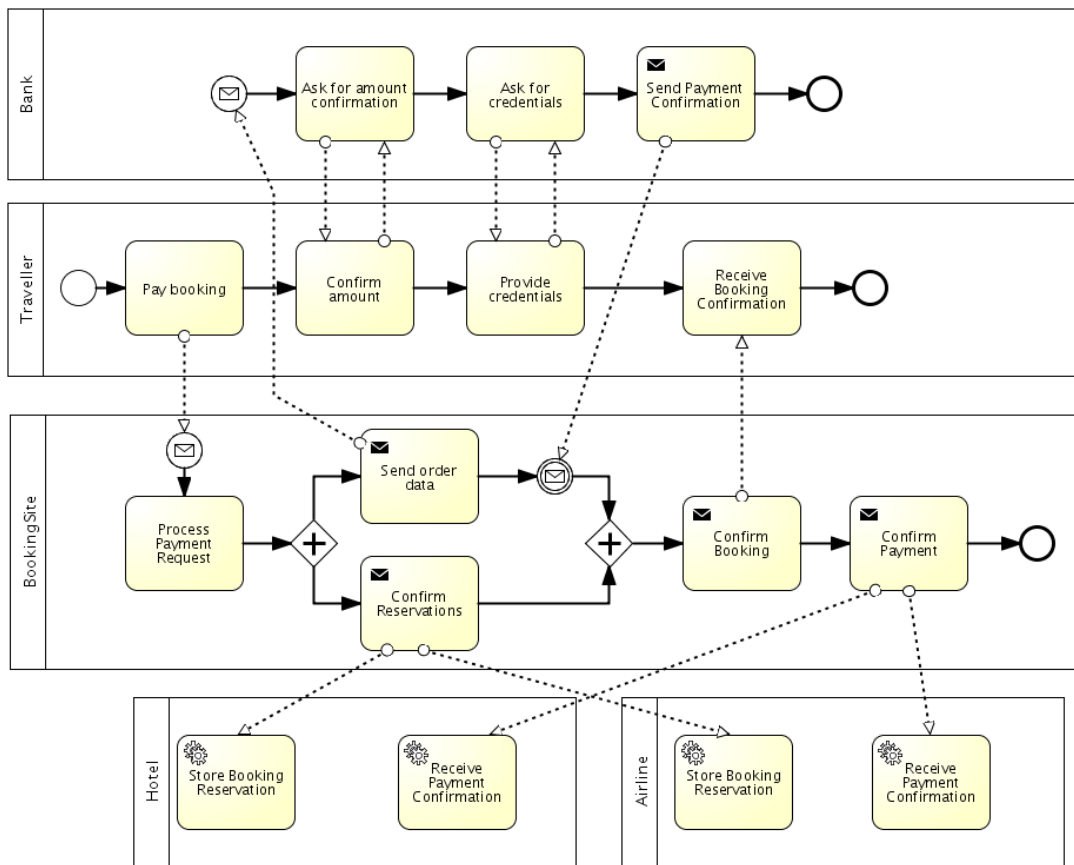


Figure 5.12: BPMN Process detailing the choreography in figure 5.10 on page 39 for payment and booking confirmations

Table 5.1: Scoring of requirements for service modelling in BPMN 2.0

Service modelling requirement	Score
Role and Entity distinction	2
Reusable service model elements	3
Service interaction protocol	4
Service provider/consumer	1
Initiator of a service	4
Interface definition	3
Composite structures	5

Role and Entity distinction is available in BPMN 2.0 through the reference from Participant to PartnerEntity and PartnerRole. The reference is many to one, meaning a Participant can only be one Role at a time. The fact that a Participant only represents one entity makes sense, since an entity is a specific object, while a role has a many-to-many-relationship with an entity (3.4.2 on page 15). BPMN also has no graphical distinction between an entity and a role, meaning that a modeller need to use naming conventions to separate the two. The fact that one ChoreographyTask can be used in several Conversations brings light upon a problem here, because you might want to model Roles in a Choreography but entities in the Conversation, but there is no way to connect these together. The use of multiple roles for one participant is consequently vague.

Reusable service model elements requirement is partially available in BPMN 2.0. If we think of a Choreography as a Service, a Choreography can reference a Conversation, but not the other way around. They are linked implicit through the use of MessageFlow and the CorrelationKey system and Choreography links to a Conversation via the ConversationAssosiation containment. This means that a Conversation diagram does not have an explicit link to a Choreography or a ChoreographyTask. Also, the ParticipantAssosiation is defined to be used mainly in cases where there exists some type of CallActivity in a process thus associating two participants, one in the calling process and one in the called process. There are many ways to connect diagrams to each other, but they exist mainly on a meta level and cannot be diagrammatically created.

Service interaction protocol is the sequence of interaction that need to take place between two or more participants. The protocol needs to contain some task descriptions, a direction and contact points list, i.e. interfaces. The Choreography diagram in BPMN 2.0 provides these modelling capabilities. The diagram provides a clear direction through use of SequenceFlow and associates Participants implicit with each other through the MessageFlow lists. The connection of the interfaces is done on a meta model level behind the scenes, so what each task is performing is down to the naming convention. Since a ChoreographyTask typically has one-to-many MessageFlows, the granularity of the protocol is not known. It could be a simple web service request-response task, or it could be a complete booking happening.

Service provider/consumer is not available as a distinction in BPMN 2.0. A service always has a provider and a consumer, who is giving value and who is receiving value. In BPMN 2.0 Choreography there is a distinction on who is the initiator of the ChoreographyTask. This indicates who is providing the service (The not initiating part), but not a specification of such. In a hotel booking protocol, for instance, there are many tasks being performed and both hotel and hotel customer initiates tasks. But only the hotel is a provider of the service. The Conversation diagram has no indication on who is providing service in the Conversation.

Initiator of a service is the one that takes the initiative to engage in service providing. The ChoreographyTasks always has an initiator. The initiator of a task is the one that sends the first message. The Conversation diagram has not any indications on who is initiating an interaction. The Collaboration diagram has this defined implicit because we see the actual messages.

Interface definition is the specification of an interface provided publicly to any consumer. What tasks can be performed. There are two ways to define this in BPMN 2.0. One is to define a Service with an interface with a set of operations. There exists no diagram for this in BPMN. As a modeller you can then hook operations to a ServiceTask contained in the participant that references the interface with the operations. This is defined in BPMN as a machine-to-machine communication only. The other way to do this is implicit by defining a participant with many message ReceiveTask and SendTask. These provides public access points that can be used as an interface definition for human-to-human and human-to-machine interaction. In addition there are ManualTask, but that is out of scope for this discussion.

Composite structures are the structures where a system uses other systems to provide some output. Examples are a participant providing a booking service. To provide a booking service it needs to use other participants or structures in order to provide the service. This could be modelled as elements inside other elements or implicit through some type of association patterns. In BPMN 2.0 this is possible in several ways. A Conversation diagram can have sub-conversation diagrams and a Choreography can have sub choreographies. But the most powerful way is by means of the CallActivity functions (20, p. 95-97). A task can call some other public, or global, process and provide data in and return data back again.

5.1.4.2 Information requirement evaluation

As a general comment, BPMN is a business process modelling language, and it is expected to rate quite low on these requirements. BPMN restricts the definitions of data types and messages into some string of data, i.e. some XML based information model defined by a XML schema (SOAP messages is one example). BPMN does therefore not provide this capability on its own.

Definition of concepts is the ability to define classes or interfaces that explains some entity, role, system or other concept. An interface is a concept of a

Table 5.2: Scoring of requirements for information modelling in BPMN 2.0

Information modelling requirement	Score
Definition of concepts	2
Definition of relationships	1
Definition of attributes	1
Definition of operations	3
Definition of constraints	1

barrier that can be used to explain what something can do. But BPMN does not provide any data centric class descriptions.

Definition of relationships is how classifications relate to each other, i.e. inheritance association and composition. BPMN 2.0 does not focus on this.

Definition of attributes are inherent properties of an object. BPMN 2.0 does not focus on this.

Definition of operations is done in interfaces for services, but not as a part of any data type. The concept of a operation is evident in the ServiceTask which performs a specific operation in an interface.

Definition of constraints is connected to relationships, which BPMN 2.0 does not focus on.

5.1.4.3 Process requirement evaluation

As a general comment, BPMN is expected to score very high on these requirements as this is the main focus for the language.

Table 5.3: Scoring of requirements for process modelling in BPMN 2.0

Process modelling requirement	Score
Participants and activities/tasks	5
Decision gateways	5
Event and error handling	5
Support for automated/human tasks	5
Model transformation and execution	5
Multi layer modelling	4
Public and private processes	5

Participants and activities/tasks are well covered in BPMN 2.0 Collaboration diagram.

Decision gateways are defined well with five different gateways: Exclusive, Inclusive, Parallel, Complex and EventBased gateways (20, p. 111).

Event and error handling is well defined in BPMN 2.0. With start, intermediate and end events. An Error event can contain an Error message (20, p. 104-105).

Support for automated/human tasks is well defined through activities. Separation between message tasks, service tasks human tasks, and further on. Each have different capabilities according to expected behaviour for the different task types.

Model transformation and execution is a main improvement from BPMN 1.x to BPMN 2.0. BPMN 2.0 has a MOF based meta model thus provides compatibility with OMG's MDA framework. The formalisation of the BPMN mapping to WS-BPEL is not evaluated in this discussion but is expected to work well, thus the full score.

Multi layer modelling is the ability to model clearly defined levels of granularity. BPMN 2.0 does this by utilizing three different diagram types. But get some minus points due to not having a clear definition on the granularity in the diagrams, separation of concerns and unclear bidirectional diagram linking.

Public and private processes is easy to use and understand in BPMN 2.0 Collaboration diagram.

5.2 SoaML

SoaML (Service oriented architecture Modeling Language) specification provides a meta model and a UML profile for the specification and design of services within a service-oriented architecture. It covers extensions to UML 2.1.2 to support the following new modelling capabilities (22):

- Identifying services, the requirements they are intended to fulfill, and the anticipated dependencies between them.
- Specifying services including the functional capabilities they provide, what capabilities consumers are expected to provide, the protocols or rules for using them, and the service information exchanged between consumers and providers.
- Defining service consumers and providers, what requisition and services they consume and provide, how they are connected and how the service functional capabilities are used by consumers and implemented by providers in a manner consistent with both the service specification protocols and fulfilled requirements.
- The policies for using and providing services.
- The ability to define classification schemes having aspects to support a broad range of architectural, organisational and physical partitioning schemes and constraints.
- Defining service and service usage requirements and linking them to related OMG meta models, such as the BMM course-of-action, BPDM Process, UPDM OperationalCapability and/or UML UseCase model elements they realize, support or fulfill.
- SoaML focuses on the basic service modeling concepts, and the intention is to use this as a foundation for further extensions both related to integration with other OMG meta models like BPDM and the upcoming BPMN 2.0, as well as SBVR, OSM, ODM and others.

SOA can be understood at different levels ranging from system architectural design, where how systems communicates and shares information all the way down to primitive types (like integer and string), to business- or organisational processes at a higher level. The SOA approach to architecture helps with separating the concerns of what needs to get done from how it gets done, where it gets done or who or what it does (22, p. 13).

The key concept of SoaML is the service. A service is the exchange of value from one participant to another. The knowledge of how to use the service and access to use it is provided by a service contract. In a service there exists a service provider and a service consumer, one of which is the initiator of the service interaction. A service is given by a participant and the participant, which can be any entity: a human, an organisation or a computer system.

5.2.1 Usage

SoaML provides two main approaches to Service Oriented Architecture: Service-Contract based and ServiceInterface based SOA. In addition, the modeller can use the simple interface approach.

Simple interface approach is the well known interface from object oriented language but used typically in a web service environment. There are two types of uses: Document style and RPC style. The first is a single typical data-in operation and the use of the interface do not need any protocol. The latter is more OO style with multiple input variables and return value. The interface approach is a one way communication where the service provider do not need to know the consumer (22, p. 15,65-66). This approach is still important to keep in mind when designing the interfaces and message types used in a communication.

ServiceInterface approach extends the simple interface to a bidirectional interface. The ServiceInterface is in itself an abstract construction that realizes an interface and uses another. The provided interface is the one that is realized and the interface that is used is a callback functionality used for asynchronous communication. The provider of a service defines a ServiceInterface and possibly a protocol for its use and provide this service through a ServicePort. A consumer of then adheres to this service interface with the use of the same but opposite interfaces or a set of compatible interfaces through a RequestPort (22, p. 15,20-24).

ServiceContract approach defines a contract that specify how providers and consumers work together to exchange value. The ServiceContract details roles defined as interfaces, either as Provider interface or Consumer interface, and participants then binds to these interfaces. The contract then contains the protocol for the interaction between the binded participants (22, p. 15,27-29).

“Capabilities represent an abstraction of the ability to affect change” (22, p. 40). A Capablitiy is an entity with a set of functions that a service provided by a participant might offer. It can be thought of as an interface in an object oriented programming language where a class implementing an interface must support those operations given by the interface. But it differs by giving a Participants the ability to realize a set of Capabilities through an ServiceInterface. This is added to the SoaML because there is a need for separation of concerns to what a participant is able to to, without specifying how it does it. In a sense, it is an abstract interface for an entity.

A ServiceInterface can realise a UML Interface thus providing some type of service. In addition, the ServiceInterface uses a set of other interfaces to provide the service. SoaML has the ability to model the behavior of the service, both the delivered services, or standard interface behavior, and services it might use to be able to deliver the service. This can of course be used to model the communication between two participants asynchronously (22, Figure 68, p. 120). One effect of the specification of a ServiceInterface is the fact that from which point of view you have you might have different ServiceInterfaces. This becomes apparent if you model capabilities



Figure 5.13: Participating in Service - The use of conjugate (tilde) naming for opposite service interfaces

for both a traveller and a hotel. The capability for a traveller is, say, to be able to book a room and receive an order confirmation. The capabilities for a hotel is to process a room request and send an order confirmation. This yield two different ServiceInterfaces. In the SoaML specification these instances can be modeled by means of ports on participants where one is stereotyped as Service and the other is stereotyped as Request. SoaML then gives the naming convention with a tilde (~) on one of the ports and the modeler gives the same names on both sides. This means that they are opposite functions fulfilling each other (22, p. 24). See figure 5.13 for an example from SoaML specification which uses the equivalent dealer / manufacturer example with conjugate ServiceInterfaces.

UML 2 delivers two types of diagrams for service interaction specification. The Collaboration diagram and the Port Composite diagram from composite structure package (19). SoaML can use both diagram types, and the choice of which to use is a decision to be made by the model developer and the case. The ServiceInterface based approach using the port-composite diagram is typically chosen if there exists services that are bound to any participants (22, p. 40). The ServiceContract approach separates the contract from the participant and makes them reusable (22, Figure 9, 10, p. 26-27). Both approaches can be used separately or in conjunction with each other.

5.2.2 Travel case - SoaML

The travel case will be modelled differently in SoaML than in BPMN. SoaML, as discussed, is more based on the service oriented architecture approach while BPMN is, as the name applies, a business process oriented language. This means that SoaML provides the ability to create both the participant architecture, the sequence of interaction for some protocol, all the way down to the classes and interfaces defining the data and operations.

For the travel case, I will be using a top-down approach. The reason for this choice is based on the idea that by identifying roles for participants and ServiceContracts before specifying any implementation specific models is a more dynamic methodology since we start from scratch. It is easier to change a CIM model than a PIM

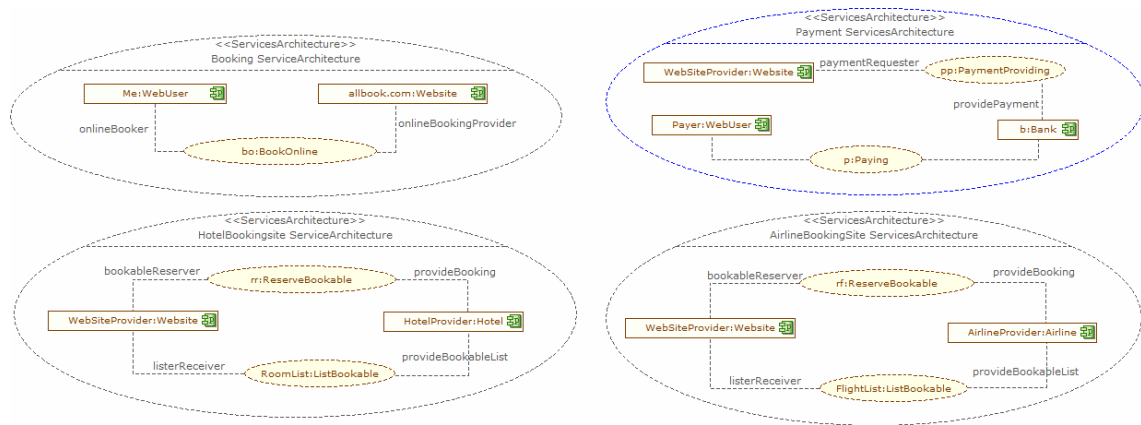


Figure 5.14: SoAML services architecture view of the travel case

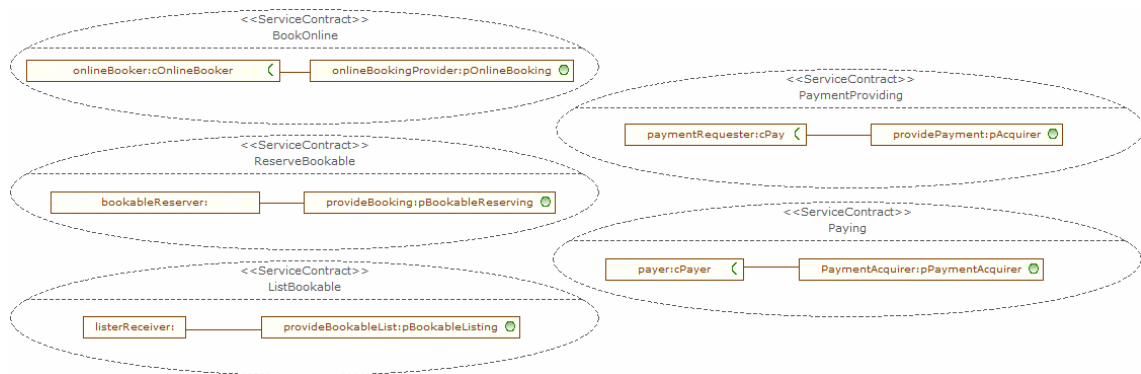


Figure 5.15: SoAML service contract view of the travel case

or PSM model, and the changing of a CIM model propagates more changes downward than the other way around which makes the work done on a high level CIM even more important. It is not clear where the boundaries of CIM and PIM level models exist in SoAML. In addition, the SoAML model will be two fold separating ServiceContract based SOA from the ServiceInterfacebased SOA.

5.2.2.1 ServiceContract based SoAML

When doing top down modelling in SoAML you start by defining ServicesArchitectures comprising of roles typed as participants. These roles are bound together using the ServiceContracts. A ServiceContract is the binding acknowledgement between participants or roles that they will communicate and provide the means for this communication. Let us start by discussing the ServicesArchitectures found in figure 5.14. There are four defined ServicesArchitectures here. They could all have been combined, but by separating the different interactions going on in the system, it get easier to read the model. The Booking ServicesArchitecture contain the two participant Me and allbook.com represented by the web user and web site accordingly. They have a contract that they will communicate through, namely the BookOnline service contract. In this contract the web user plays the role as onlineBooker and the web site the role of onlineBookingProvider.

The hotel booking and airline booking services architectures are very similar to each



Figure 5.16: UML interaction diagram detailing a protocol between an `onlineBooker` and an `onlineBookingProvider`

other. This is because the only difference is the participant of hotel and airline. It could be argued that they could in fact be the same architecture by utilizing the role concept more to abstract the participants as for example `Reservee`. But by doing it like this, the model is easier to change in case the hotel booking and airline booking should not be modeled the same way. But the `WebSite` is defined as communication for both the hotel and the airline through the same service contracts.

The payment services architecture uses three participants where the `WebSite` communicates with the bank, and the bank to the web user, now in the role of payer. This is because the web site must tell the bank how and with what data the payer should pay for its reservations and then contact the web user to perform the payment.

All the service contracts that exists in the different services architectures is shown in figure 5.15 on the previous page. They are generally quite similar with one provider interface and one consumer interface. But the `ReserveBookable` and `ListBookable` are different. Here the service contracts show a simple interface based contract where the consumer is not defined by any interface. The idea is that the provider of a bookable already have a web service implemented.

Each `ServiceContract` that is not based on a simple interface have an interaction diagram that details the protocol. Figure 5.16 displays such an interaction diagram for the `OnlineBooker` service contract. In this case the consumer is also implicit the initiator of the interaction.

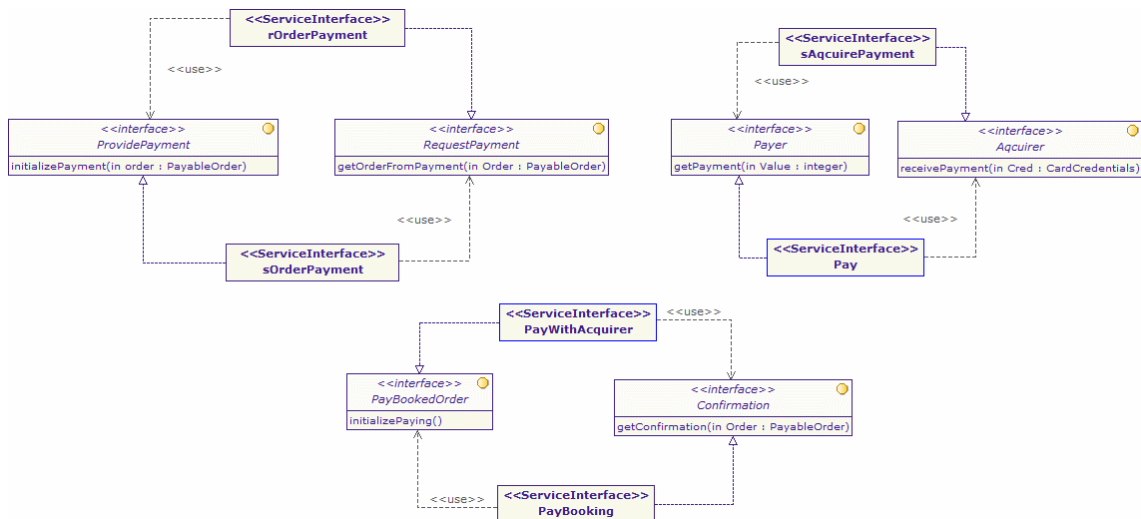


Figure 5.17: SoaML ServiceInterfaces for the payment part.

5.2.2.2 ServiceInterface based SoaML

When doing ServiceInterface based SOA, starting with a ParticipantArchitecture (top-down) seems strange. Jim Amsden say that SoaML is a bottom up modelling language (1), and referred in his discussion to ServiceInterface based SOA. These models are consequently created starting with defining the ServiceInterfaces, then the interfaces and lastly adding the ports to the Participants before creating the ParticipantArchitecture. (ServiceContract based SOA is top-down approach.)

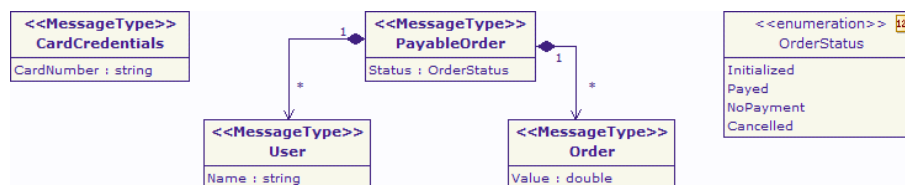


Figure 5.18: SoaML MessageTypes for the interfaces' operations

Figure 5.17 displays the created ServiceInterfaces. There are three sets of service interface definitions and each set has two sets of interfaces and ServiceInterfaces. The interfaces have one operation each for simplicity and they are based on document style interface definitions. The input parameter are typed as MessageType as depicted in figure 5.18. The MessageTypes are self explanatory.

Figure 5.19 on the next page show the ParticipantArchitecture. The figure 5.19 on the following page displays the participants on the left with all the service interface ports with the provided and required interfaces. On the right there is an abstract participant that combine all the interfaces in the port composite diagram.

The diagrams are rather self explanatory, but there is an interesting distinction. The web user has two ports both of which are request ports. In the case of the interaction between the bank as the acquirer and the web user as a payer, the bank is initiating the interaction and is also providing the service. This means that it is the bank that contacts the web user not asking to get the payment, but asking the

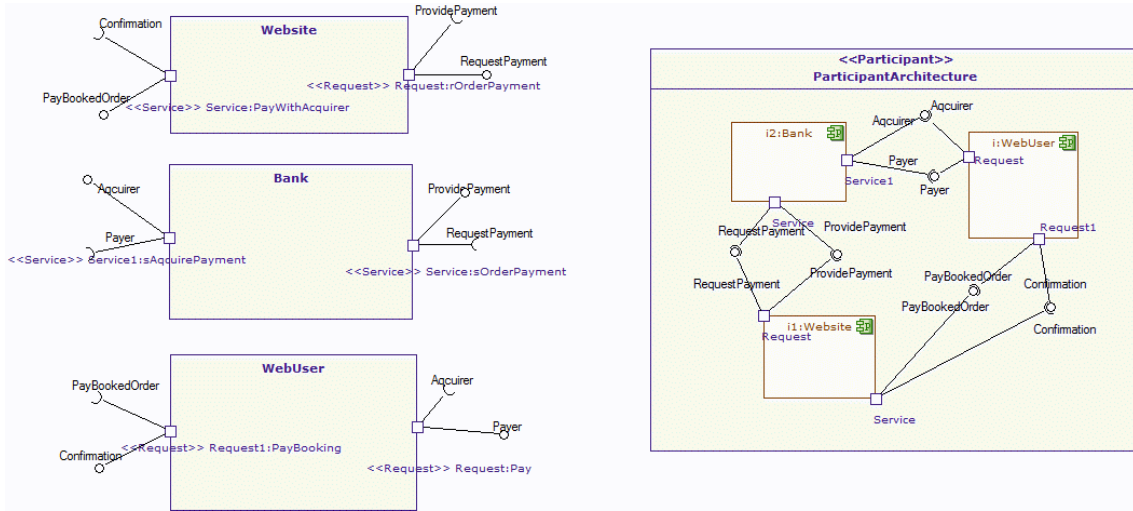


Figure 5.19: Two participant architectures with ServiceInterface ports

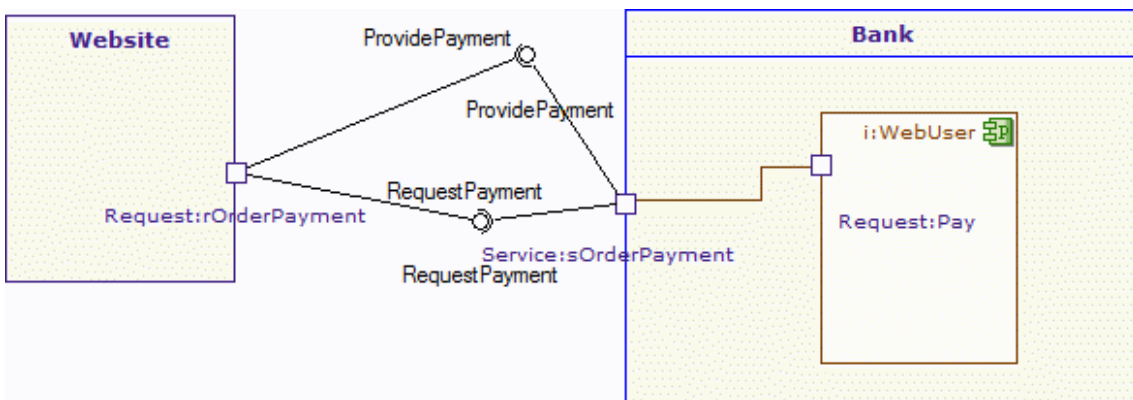


Figure 5.20: Two participant architectures with ServiceInterface ports

web user to pay through the provided service. This example shows that there needs to be a defined provider and consumer in addition to an initiator of the interaction.

Figure 5.20 on the preceding page show a version where the bank is providing a service through a port. In this small example the web site only provides data on “who” is paying “what”, and the bank provides service by contacting the web user.

5.2.2.3 Capabilities

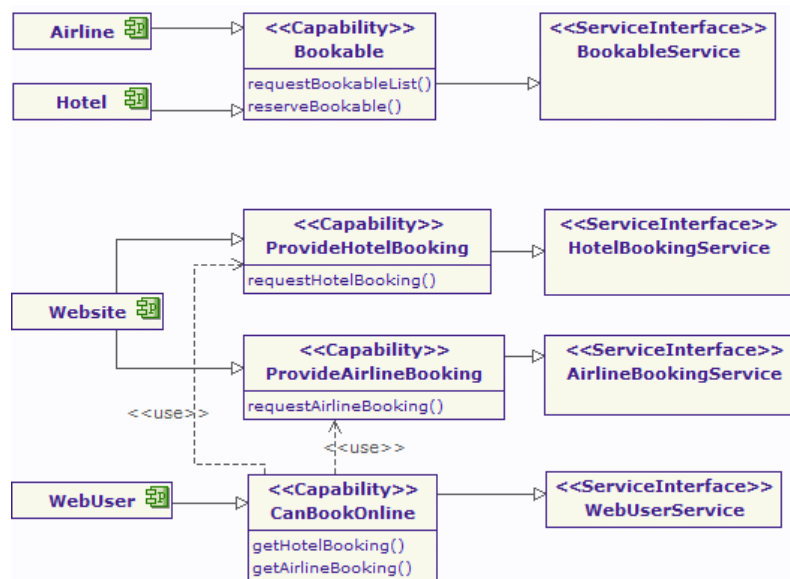


Figure 5.21: SoAML participant capability view with ServiceInterfaces

Figure 5.21 shows all four participants involved in the booking part of our process: Airline, Hotel, WebUser and WebSite. We see that both Airline and Hotel realise the same Bookable capability. This means that both provide some type of bookable service. WebSite provides two types of capabilities namely the provideHotelBooking and provideAirlineBooking. These capabilities are named differently for semantic purposes. The web user has the capability to book online. This capability is thought of as a human ability, e.g. able to use web pages, but that distinction is not included in the diagram. All capabilities realise one service interface.

5.2.3 Requirement evaluation

There are three focus areas of the requirements for a modelling language outlined in section 4.3 on page 23: Service, Information and Process modelling.

5.2.3.1 Service requirement evaluation

SoAML is designed to be a service architecture oriented modelling language, and focuses greatly on the description and specification of a service. Because of this, SoAML is expected to score high on these requirements.

Table 5.4: Scoring of requirements for service modelling in SoaML

Service modelling requirement	Score
Role and Entity distinction	4
Reusable service model elements	5
Service interaction protocol	4
Service provider/consumer	5
Initiator of a service	3
Interface definition	5
Composite structures	5

Role and Entity distinction is evident in the ServiceContract based SOA. The concept of an interface playing a role in a ServiceContract, and by binding a participant or a role to the ServiceContract role, the participant then play that role. A participant can play many roles in one ServiceArchitecture. SoaML does not get full score, since the ServiceInterface based SOA does not incorporate the role concept.

Reusable service model elements. A ServiceContract and a ServiceInterface is easy to reuse for several participants and the separation of concerns that exists in SoaML encourage reuse of the service model elements.

Service interaction protocol is provided through use of UML interaction diagram or activity diagram. SoaML specification is not clear on how to combine the protocols in a business process.

Service provider/consumer is available in both ServiceContract based and ServiceInterface based SoaML. In a ServiceContract this is available with a stereotyped interface (Consumer, Provider). In ServiceInterface based SOA, this is done with the stereotype of Request or Service on the port element.

Initiator of a service is implicitly defined through the interaction protocol for a service, but there exist no diagram hint on who the initiator of an interaction is.

Interface definition. SoaML uses the UML interface classifier to define an interface. The interface can have operations. In addition, SoaML has constructions for bidirectional interfaces further extending the interface from a one way communication to an asynchronous conversation.

Composite structures. UML Collaborations can have composite collaborations through the collaborationUse type. In the port-composite structures, participants can model internal constructions or protocols as composite structures.

5.2.3.2 Information requirement evaluation

UML is the de facto standard for information modelling, and SoaML, being an extension to UML, score expextedly high on these requirements.

Table 5.5: Scoring of requirements for information modelling in SoAML scores

Information modelling requirement	Score
Definition of concepts	5
Definition of relationships	5
Definition of attributes	5
Definition of operations	5
Definition of constraints	5

Definition of concepts is defined through classifier type. Class, Interface, DataType extends the Classifier type (19). SoAML has the MessageType stereotype to distinguish constructs used in web service descriptions and web service communication, like WSDL and SOAP.

Definition of relationships. Aggregation, association, inheritance are types of relationships possible in UML.

Definition of attributes are possible for all DataTypes and Class constructs and all typed elements can be attributes.

Definition of operations are possible on Interface and Class.

Definition of constraints. Associations and aggregation have cardinality on both ends of the association. “OCL supplements UML by providing expressions that have neither the ambiguities of natural language nor the inherent difficulty of using complex mathematics.³”. UML gets full score even though OCL provide it with the last piece of the puzzle for constraints in a model.

5.2.3.3 Process requirement evaluation

SoAML does not extend UML process modelling diagrams: The activity diagram or the interaction diagram. Consequently, SoAML has the same capabilities as UML in process modelling. The SoAML specification discuss how these diagrams can be used for protocol descriptions, but it do not focus on business process modelling (22).

Table 5.6: Scoring of requirements for process modelling in SoAML scores

Process modelling requirement	Score
Participants and activities/tasks	4
Decision gateways	3
Event and error handling	4
Support for automated/human tasks	2
Model transformation and execution	4
Multi layer modelling	3
Public and private processes	3

³http://en.wikipedia.org/wiki/Object_Constraint_Language

Participants and activities/tasks are available in the activity diagram of UML.

The use of both interaction diagram and activity diagram in SoAML is for protocols of ServiceInterface and ServiceContract. Therefore the interaction is specified between mainly interfaces, according to the SoAML specification.

Decision gateways are available in UML activity diagram.

Event and error handling is only available in the activity diagram from UML.

The use of events and errors is not discussed in the SoAML specification.

Support for automated/human tasks. SoAML does not differ between a human and a machine task formally. A participant can be all participants that can engage themselves in a service exchange and consequently the distinction between human and machine must be defined semantically.

Model transformation and execution. UML is based on MOF and thus is in compliance with OMG MDA for model transformation capabilities. Model execution is not a main focus for SoAML and is not covered in the SoAML specification. But through a model driven architecture, execution will be possible.

Multi layer modelling. In both ServiceInterface and ServiceContract based SoAML, there are three levels of modelling; The participant and their connection points, the specification of the connection points and lastly the protocol. The actual process is only specified in a protocol meaning that there is only one level of process modelling, although there are two diagrams.

Public and private processes. A process in SoAML is tightly connected to the protocols of either the ServiceInterface or the ServiceContract and a participant of an interaction is expected to implement this protocol exact thus the protocol must be public. SoAML focuses very little on how a process task is executed inside a participant but can model the private architecture in a ParticipantArchitecture.

5.3 Conclusions based on the two evaluations

To sum up the evaluation of BPMN 2.0 and SoaML the required scores are brought together and the results are available in table 5.7. We can see that SoaML does better in both service modelling and information modelling. BPMN 2.0 scores better in process modelling. The results are as expected as they do well in the area they are designed for.

Table 5.7: Comparing requirement scores for BPMN 2.0 and SoaML

Requirement description	BPMN 2.0	SoaML
Service model		
Role and Entity distinction	2	4
Reusable service model elements	3	5
Service interaction protocol	4	4
Service provider/consumer	1	5
Initiator of a service	4	3
Interface definition	3	5
Composite structures	5	5
SUM	22	31
Information model		
Definition of concepts	2	5
Definition of relationships	1	5
Definition of attributes	1	5
Definition of operations	3	5
Definition of constraints	1	5
SUM	8	25
Process model		
Participants and activities/tasks	5	4
Decision gateways	5	3
Event and error handling	5	4
Support for automated/human tasks	5	2
Model transformation and execution	5	4
Multi layer modelling	4	3
Public and private processes	5	3
SUM	34	23
SUM overall	64	79

This corresponds to the models created in both languages. Process modelling in SoaML is not well defined and seems much weaker than in BPMN 2.0. BPMN 2.0 were not able to model the data for message exchange at all which is quite well defined in SoaML. Both languages do well in service modelling, but SoaML do better overall. The difference is not that strong. The only reason why BPMN 2.0 do worse is the lack of role modelling capability, the virtually non existent provider-consumer distinction and the ability to clearly define interfaces and the use of them.

The evaluation answers the research questions (see section 1.2 on page 2) of what

the languages cover in terms of capabilities and in what degree they bring solutions to the problem domain. We have also seen the strengths and the weaknesses. What this language evaluation do not provide is answers to the question of overlap between languages, and how these overlaps can be combined. The evaluation does provide hints to where these langauges are overlapping, but not in detail. The next chapter will present a vision for BPMN4SOA and try to answer the last three research questions.

BPMN4SOA vision

The business process modeling discipline and the service oriented architecture are two different concepts. While BPM traditionally targets people and their work, SOA targets systems architecture. “However, there is also widespread usage of the terms BPM and SOA interchangeably”(15, p. 16). As seen, SoaML and BPMN 2.0 are complementary. Although they overlap in some areas, they are intended for use in different modelling arenas. BPMN 2.0 provides a design for creating BPM models used by some entity to reach its goals. The model is created by business analysts that potentially do not know anything about mapping systems, persisting data or execution frameworks. Their task is to create a model suitable to the business for documentation or execution. SoaML on the other hand is a modeling language providing an outward facing view of entities, their responsibilities to each other and capabilities they might provide (1). SoaML has diagrams for modelling behavior, but being a UML profile it is a more technical modelling language than BPMN. Both languages can be argued of being the two sides of the same coin. Both implements and use services, and both can be applied to the technical and to the business level.

I will propose a new language called BPMN4SOA which utilize the strengths from both languages. The vision for BPMN4SOA is to make a business process language for use in the SOA paradigm that incorporates service modelling and information modelling.

To create a completely new specification is not the way forward. But instead, build on something that already exists. BPMN 2.0 or SoaML is a natural choice for this thesis. But which one is the best suited and how to utilize them to reach the goals? An evaluation of the similarities of the notation of both BPMN 2.0 and SoaML might provide a formal mapping uncovering the best solution for BPMN4SOA.

6.1 Mapping and integration of SoaML to BPMN 2.0 notation

BPMN 2.0 is a huge leap from BPMN 1.x in terms of executability. While 1.x specification does not have any formal mapping to the execution language BPEL, BPMN 2.0 does. This ability built in to the 2.0 version puts more burden on the modeller to think in terms of system execution. Some say BPMN 2.0 is getting too complicated too fast. To address this, Bruce Silver proposes a three level modeling methodology for BPMN 2.0 modelling: Descriptive BPMN, analytical BPMN and executable BPMN (27).

Descriptive level focuses on simplicity. Describing the complex world in its simplest forms. It does it by limiting the palette of shapes used in the modelling process, and by doing that the modeller is forced to think top down without compromising the process flow or semantics used in the flow. This helps all participants in the modelling process to understand and discuss the model.

Analytical level extends the descriptive level by focusing on event handling and error handling and other patterns that might occur in the process; It takes the model downward to a more technical meaning by utilizing all shapes defined in the standard.

Executable level is where the mapping to the execution is addressed. Adding data types, defining human tasks vs. computerized tasks, mapping to services are examples of focus area of this last level. This level is not addressed in BPMN 1.x and is one of the new features of the 2.0 version.

The tree level method to BPMN modelling is not a part of the BPMN 2.0 specification. Limiting the complexity of the modelling process to fit different levels of expertise, from business educated to the computer educated, can help limiting the communication gap that seems to exist. The idea of dividing the modelling methodology into levels can be very useful for the mapping between SoaML and BPMN. By doing the same limitations in the descriptive level 1 many complex constructs, like compensation events and error events, can be left out of the discussion (27).

Jim Amsden proposes three approaches to integrate SoaML and BPMN, and all three have different pros and cons: 1) None; 2) model-to-model mapping; 3) meta model integration. What he means by “None”, is that we assume that the tools for modeling are independent and that the overlaps are intentional, basically leaving the integration up to tool vendors. This approach does not benefit from complementary capabilities that might be present in the two languages. Thus, “None” will not be a part of this discussion. The “model-to-model” approach focuses specifically on the complementary capabilities, but this results in redundancy of model elements in both languages. There are many unique features to both that does not map easily making a “model-to-model” mapping lossy. The last approach is the meta model integration. Basically, it is adding the BPMN meta model to SoaML meta model, or vice versa by mapping similar elements directly and adding the redundant elements to the target meta model. This method provides the best integration, but basically

is a new language specification with all the problems and benefits that follows the standardisation work (1).

The method used here is the model-to-model mapping: Trying to represent a SoaML model as good as possible in BPMN 2.0, and discuss how good it fits.

6.2 SoaML ServicesArchitecture

The background for this mapping between the BPMN 2.0 Conversation diagram to SoaML ServicesArchitecture diagram is feedback from systems architects and business process modellers to the SoaML standardisation committee that the notation is difficult to understand and use, and that the BPMN notation is easier to understand. There are many similarities with the notations and meta model, but the look and feel is quite different.

Table 6.1 on page 63 on the second line show one ServicesArchitecture and one Conversation. The similarities is quite obvious with participants and the communication through one element. But BPMN 2.0 lacks what SoaML has, and it is the role specification on each binding to the ServiceContract. We need to find how SoaML Participant and ServiceContract maps to BPMN.

6.2.1 Participants

Both BPMN 2.0 and SoaML have the concept of a Participant. Both can be used at different abstractions; corporation entities, corporation departments or computer systems are examples of uses for the Participant element. In SoaML a participant is a UML stereotype and extends the UML class element. In BPMN 2.0 the Participant is represented by a Pool and can have a EntityName or a RoleName. A difference between BPMN and SoaML is that a Pool also can have lanes where lanes can be thought of being departments under the main Participant. There are suggestions for lanes to be represented by the SoaML Ports which is the request port and service port of the SoaML Participant (1).

In BPMN Collaboration diagram, Participants are represented by the element Pool that also can contain lanes. Both the Pool and the Lane can contain orchestration processes. However, in the Conversation diagram, Participants are represented by a simple box that do not contain anything in the diagram itself (20, p. 333). Participants in BPMN can only exchange MessageFlow, optionally containing a Message, between each other. This flow defines that there are communications between the participant and the message is the data, or physical entity, being sent (20, p. 113-123).

In SoaML the Participant is, as described above, a Stereotype of UML class with its own meta model. A Participant can represent the same entities as the BPMN Participant. A Participant may have ports, defined as the stereotypes Request and Service, which are the interaction points for the services are either consumed or

offered. A Participant can also contain a composite structure for its sub-components. In a collaboration, participants communicate through a ServiceContract binding to a role in the contract (22, p. 68).

To sum up, the BPMN Participant (Pool) can send and receive Messages via the MessageFlow element to other participants. In a Collaboration diagram, a Participant has processes, but in Conversation diagram they do not. A SoaML Participant use ports to define the interaction points between each other, and can contain a sub process. A Participant in SoaML and in BPMN is the same type of abstraction. It is a part of an interaction defined by the modeler according to the needs of the specific task at hand. A Participant can be everything that plays a role in a collaboration e.g. a person or a company or a computer system.

6.2.1.1 Service Architectures vs. Conversations



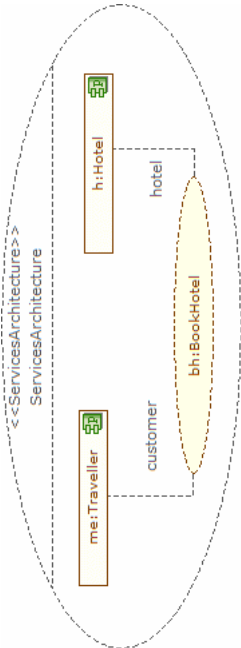
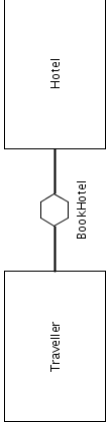
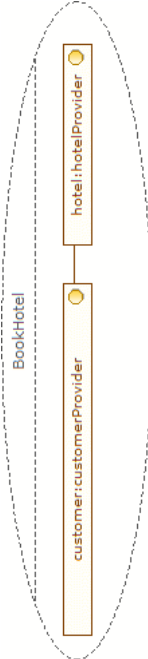
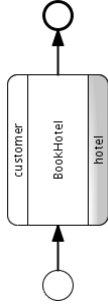
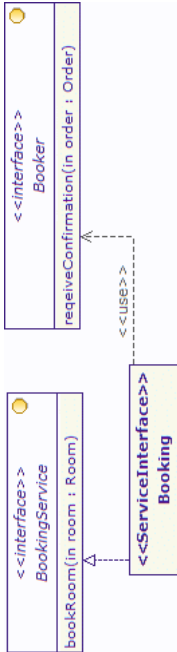
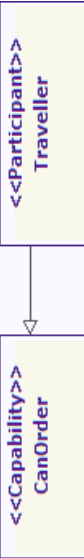
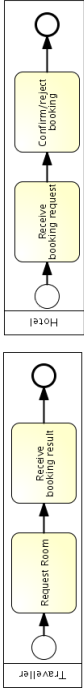
In SoaML a ServiceArchitecture is a specification of dependencies between participants through a ServiceContract specifying a binding acknowledgement for interaction by the participants. The roles played by a participant in the ServiceContract are shown in the line connecting the participant to the ServiceContract. The BPMN Conversation shows that there is a communication between the participants happening, but we can not add more data to this diagram. The role concept does therefore not map from the SoaML ServiceArchitecture to a BPMN Conversation diagram. However, the BPMN meta model has the concept of role for a Participant. Thus, the diagram can hold the meta data of a role as an attribute of the participant. The difference is that in SoaML, you can use the abstract Role or the construct Participant interchangeably in the ServiceArchitecture. Note that the Role of a Participant in BPMN does not have any graphical representation other than showing the name of either the role name or the participant name in the box.

The use of a Conversation in BPMN is not clearly defined other than it is a bird's eye view of the whole process chain. The diagram is very simple, using only participants and conversations. How to use the diagram for modelling purposes is therefore not as straight forward as it seemed at first, especially not in combination with other diagram types that are needed to represent the ServiceContract as Choreographies.

6.2.1.2 ServiceContract and ChoreographyTask

A ServiceContract is a specification and a binding contract between two or more roles. The contract specifies the roles played in the message exchange. In a ServiceContract, the roles can be represented by participant, service interfaces or interfaces. In the SoaML specification the interface is used, specifying the operations by which the communication will use. BPMN needs to use the Choreography diagram to represent this contract. A ChoreographyTask is a type of business contract. A ChoreographyTask can contain a set of MessageFlows that makes up the actual conversation between the participants. The white participant is the initiating part. A ChoreographyTask can through ConversationAssociation map to a Conversation, meaning that there can be a link between a Participant in a Conversation diagram and a Participant in a ChoreographyTask.

Table 6.1: SoaML to BPMN mapping equivalent

SoaML	Comment	BPMN 2.0
	<p>Participant is the same construct in both languages in terms of semantics and abstraction level.</p>	
	<p>ServiceArchitecture and Conversation show basically the same information, but Conversation lacks the Role element defined by a ServiceContract.</p>	
	<p>The participants on the Choreography must have interfaces to be able to map all elements.</p>	
	<p>The concept of a ServiceInterface does not map directly to any BPMN constructs, but is the interaction between two participants.</p>	<p>See figure 6.2 on page 65</p>
	<p>There is no BPMN construct to match the CanBook capability directly, but tasks in a participant as public process might model the same construction.</p>	

In the ServiceContract diagram and in a ChoreographyTask there are differences in meta information. The SoaML diagram contains interface through which the binding participants communicates. The BPMN diagram can contain the same information by giving the participant an interface, but the presence of this interface is not displayed as a part of the diagram of BPMN. In addition, the Choreography-Task contains the information of who is the initiator of the interaction. This is not included in the SoaML counterpart example, but can be by adding the stereotype of Provider and Consumer to the interfaces in the ServiceContract. However, the consumer is not always the initiator, as seen in the SoaML case study above. Therefore, the initiator of an interaction in a ServiceContract is specified implicit in the protocol for the contract. The mapping between these concepts are very good, but there are some differences.

6.3 SoaML Participant Architecture

The ParticipantArchitecture model participants by means of the port-composite structure diagram in UML. Participants communicates through ports typed as a ServiceInterface. Ports are also stereotyped as either Request or Service. This is very similar to the ServiceContract.

6.3.1 Service Interfaces

A ServiceInterface is an abstraction in SoaML of the interaction between two service providers. In a normal web service commonly used today, the interface is used to describe the web service. One participant, the consumer, requests a web service, the provider, on a specific operation, and gets a return answer. The simple Interface style is in a sense a one way communication. However in a ServiceInterface, the provider uses another interface to provide its service and thus is a two way communication.

This can be used in two ways; either by the provider service consumes a different service to provide its service, or the service can use the initial consumer. In the SoaML ServiceInterface part of table 6.1 on the preceding page there is a ServiceInterface that uses one Interface and realizes another Interface. If the initiator or consumer in the ServiceContract specifies the interaction is the same as the one that provides the Booker interface, then the ServiceInterface can be represented by a simple public process shown in BPMN figure in the mapping table. In that model, the ServiceInterface is actually the two messages going from one participant to another together with the message going back again. This is because the second message is from another task than the first message, meaning it is something different than a regular RPC happening in this case, i.e. both participants know how to contact each other.

The section on SoaML 5.2 on page 46 shows that two participants interacting together make two ServiceInterfaces, one Request and one Service. This example shows that a ServiceInterface can not directly be mapped against the example shown in figure 5.3 on page 31. Jim Amsden proposes a mapping between SoaML ServiceInterface and BPMN: “A Communication in a Communication diagram, including

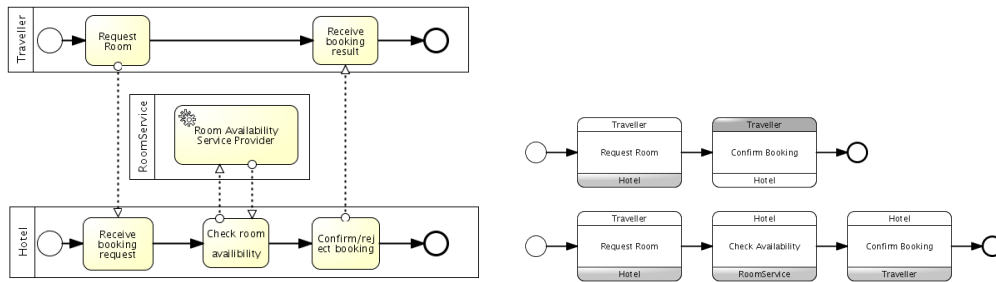
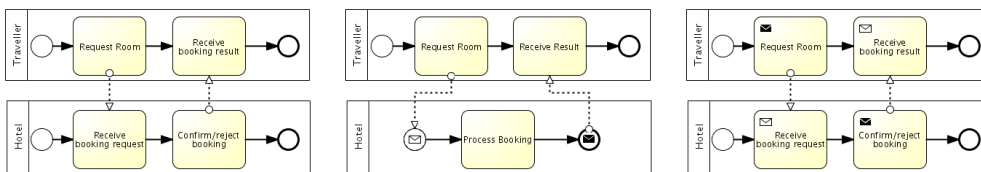


Figure 6.1: Process with two communications and choreographies representing one ServiceInterface and one simple interface



(a) human-human process (b) human-machine process (c) machine-machine process

Figure 6.2: Model alternatives of a BPMN process that can be thought of as a SoaML ServiceInterface

the corresponding messages in a communication diagram, and the choreography of those messages in a choreography diagram” (1). The communication is the example discussed above saying that there is a message exchange going on both ways. And, according to Amsden (1), you need a Choreography to specify a direction for the communication and the instantiating participant. The Choreography also separates and groups different MessageFlows together and makes it possible to define which communications are linked to each other. This means you need two different BPMN diagrams to represent one SoaML ServiceInterface.

Figure 6.1 shows a process that have three participants; The traveller, the hotel and a service the hotel uses to check the availability of rooms in the hotel (RoomService). The dialogue between Hotel and RoomService is in itself not a ServiceInterface, if we think in terms of the mapping discussed above, because the communication between Hotel and RoomService most likely is a simple interface based communication. The only task in RoomService is a ServiceTask that provides the use of a single operation specified in an interface owned by the RoomService participant. This example shows that just by adding one participant to the process and modifying the communication slightly, the mapping between a SoaML ServiceInterface and a BPMN process pattern becomes fuzzy and unclear.

SoaML and BPMN do not differentiate between human- and computer based participants. However, BPMN differentiate between human tasks and computer tasks, which SoaML do not. For instance, by using ReceiveTask and SendTask BPMN will give us the option to specify operations on the tasks to be used for the send or receive functionality. We can therefore consider the process in figure 6.2b and figure 6.2c as two different specialisations of figure 6.2a. The first restricts the process to a human-machine process and the latter restricts it to a machine-machine

process. The choice of restricting to human is not considered as a strong restriction, because the model does not specify human interaction. But since the send/receive task/events do restrict to machines, we can consider the none specified task as human for the sake of argument. There are more combinations that can be made from the constructs, but these three shows a good enough variety.

6.3.2 Capabilities

A Capability is an abstraction of the ability to affect change that a Participant can provide or a ServiceInterface can expose through a dependency. A Capability is not meant to be an exact specification on how this ability is to be used by others or how a participant implements this ability, but merely exposes the semantics of this ability so that others can understand what it is. The reason for this construct to exist in SoaML is that in the analysis process of a Service Oriented Architecture, the participants in some cases might not be known. In those cases it makes sense to model the participant's abilities rather than model the participant itself, resulting in a model of something that can be realized by the participant when it is identified. BPMN 2.0 does not provide this type of construct. Instead you can model this in BPMN 2.0 on the descriptive level for instance as a public process. But there is no formal way to distinguish between a public process as either a capability or an activity diagram, to name two examples. In other words, there is no apparent way in BPMN to specify this abstraction without conflicting other mappings.

6.3.3 MessageType

SoaML provides, as mentioned earlier in the SoaML evaluation, the ability to model the data used in communication between participants. BPMN 2.0 does not have these concepts. Thus there can be no mapping between the MessageType stereotype in SoaML against any BPMN 2.0 constructs.

6.4 Evaluation of mapping between BPMN 2.0 and SoaML

Mapping and combining the process centric BPMN and the architecture centric SoaML models are not a straight forward task. Many of the concepts maps very well if we constrain the use of several concepts in the BPMN model.

A BPMN process orchestration has no constraints in the usage of participants, tasks or MessageFlow which are the three main elements in a SOA usage of BPMN. Participants are the service providers, the tasks are the service ports and the MessageFlow is the contract between them. These three elements are combined into one in the ChoreographyTask. By adding the sequence flow in each process the protocol is defined also, and would be the same in a ChoreographyTask sequence contained as references to the process diagram. This makes the ChoreographyTask an abstract view of the same process. One ChoreographyTask only say that there are messages being exchanged, not how many and, potentially, in what order are they being exchanged.

A ServiceContract in SoaML is a contract for interaction between two roles defined by either participants or interfaces. The protocol for the interaction is then modelled in a separate sequence diagram detailing the interaction sequence between the roles through these interfaces.

The ChoreographyTask like the ServiceContract does not in it self show the protocol of the interaction, but rely on a different diagram to show the interaction sequence. They differ in the sense that a ChoreographyTask only show interactions between participants, while the ServiceContract detail roles defined either by an interface or a participant. A ChoreographyTask can only reference an interface through the meta model and a participant in the task can have a role attribute with a name, but these can not be showed in the diagram. The level of granularity that you give to a ChoreographyTask is totally up to the modeller, but setting the granularity too high, will make the possibilities for the matching process too wide and the ChoreographyTask will loose the mapping against a ServiceContract defined by interfaces.

The concept of a ServiceInterface as defined in SoaML has no direct parallel construction in BPMN. The ServiceInterface defines a port on a participant through which a participant interact with other service interfaces. A ServiceInterface realise an interface and uses another interface. Thus, ServiceInterface provides a bidirectional interaction. In figure 6.2 on page 65 there are some examples of the service interface construction, where the service interface is defined by the actual interaction that happens. One problem is that there are no differences between a BPMN process that maps to a ServiceContract than a process that maps to a ServiceInterface. But the SoaML constructions are very different.

In SoaML there are two main approaches to modeling SOA; ServiceContract based and ServiceInterface based. The specification do not define which one is best at what, but rather provides different modeling techniques and approaches to a solution for the same topic. One way to remove the problem of separating ServiceContract from a ServiceInterface is to remove the whole problem in the first place.

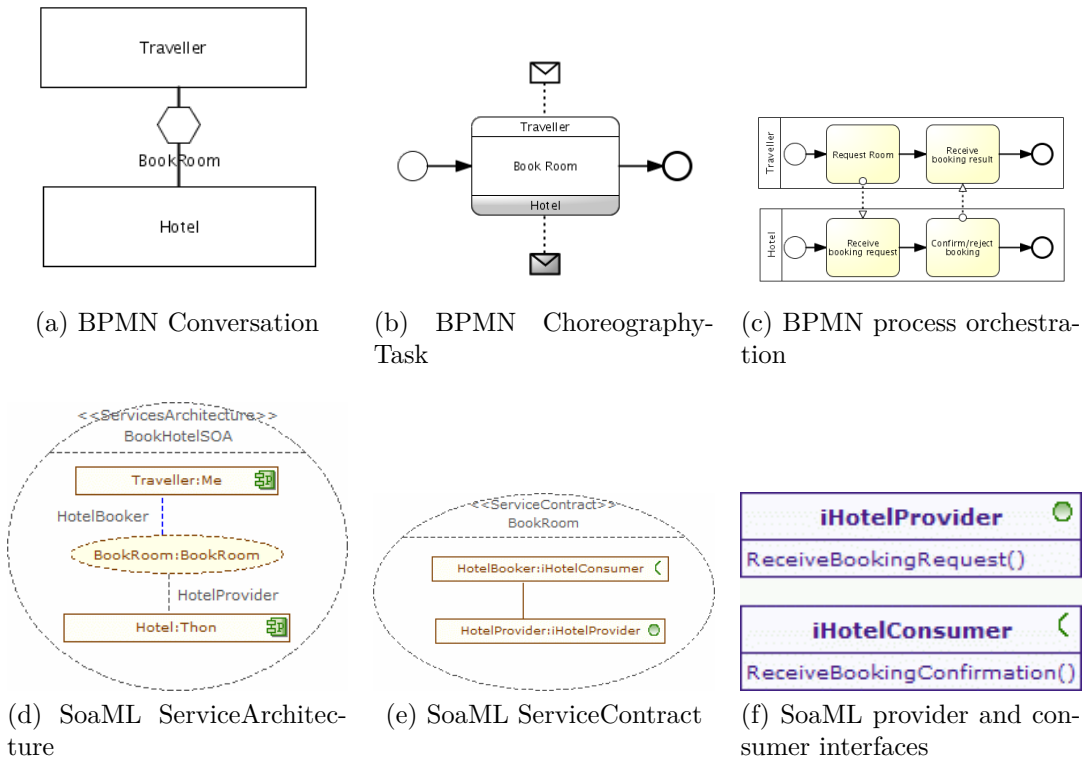


Figure 6.3: Three BPMN models for the simplest possible SOA and with supporting SoaML models for defining the interfaces

- ServiceContract and ServiceInterface approach can not be used interchangeably in the same model architecture.
- Remove the ServiceInterface approach altogether

BPMN has no notation similar to the port composite structure found in SoaML used in the ServiceInterface approach. With structures like sub conversation, sub choreographies, conversationAssosiation and participantAssosiation one might be able to model similar architectures, but it is not clear how this work in a modeling tool since all these constructs do not have any diagram notation. It could be argued that by choosing this approach, the Choreography and Conversation diagram is not needed in BPMN, and the more complex port composite diagram would be the representation for the SOA. This way the BPMN language is not given the SOA ability, but rather links two and two MessageFlows against the SOA model in SoaML.

The other outlined approach is to remove the ServiceInterface based approach altogether. The reason for this choice is simple. The notation of a ServiceContract is more similar to the notation is BPMN ChoreographyTask and the notation of a ServiceArchitecture is more similar to the notation of BPMN Conversation. Thus by adding some constrains on the modeling of the MessageFlows between participants, the ChoreographyTask can be specified through a ServiceContract adding the interfaces that needs to be used in each task.

Let us discuss a simple SOA model shown in figure 6.3. There are three constructs

from each modeling language. The first BPMN (figure 6.3a on the preceding page) model shows the Conversation between the two participants Hotel and Traveller. They communicate with each other via the BookHotel conversation. The second BPMN model (figure 6.3b on the facing page) displays the Choreography of the communication, telling us that it is the traveller who initiates the communication which consist of two messages. The third figure (figure 6.3c on the preceding page) shows the process orchestration, i.e. the direction of the conversation and tasks that are being performed. This could be expanded into a more analytical and elaborate process. The first SoaML model (figure 6.3d on the facing page) shows the ServiceArchitecture comprising of the participant “Me” playing the role of the traveller, and the participant Hotel playing the role of AvailableHotel. They communicate with each other through the ServiceContract BookRoom. This ServiceContract is specified in the next SoaML figure (figure 6.3e on the preceding page). The ServiceContract defines a contract between two roles defined as interfaces. One is the Consumer stereotype, and the other is the Provider Stereotype. The last SoaML figure (figure 6.3f on the facing page) shows the specification of these two interfaces with operations.

The Conversation diagram and ServiceArchitecture shows the same basic architecture, as mentioned above. But ServiceArchitecture also shows the roles that the participants play when fulfilling the ServiceContract. What we could have done in the BPMN Choreography is to specify the role name as the participant name. This is possible in BPMN by utilizing the ParticipantAssociation construction to bind the Choreography participants to a Conversation participants. This happens behind the scenes, so keeping the naming of participants seems to be a good practise in BPMN so others can understand the diagrams without knowing the meta data.

By constraining the communication to simple message flows as in this example is possibly a way be able to link a SoaML service architecture firmly into a BPMN process, gives BPMN the definitions of both the protocols, as UML Interaction diagrams do in SoaML, and the business process, or BPM, modelled out of the simple starting point process that are being defined here, replacing the UML Activity diagrams in SoaML.

Some constraints have now been established to the amount of MessageFlows that a ChoreographyTask should contain to represent a service. A service is the value being given from one part to another and we need the two-way communication that this provides us. The result of the constrains we give BPMN is that we define a service and service interactions implicit through this pattern as we do in SoaML.

To further SOA enable BPMN there is a need to add the concept of a role. A role is the position a class instance has in interaction with other class instances. A role is in other words something an entity has got. Another important aspect of a role is that it can be shared and reused. Being a mother is a role; not only one person can be a mother. This means that if we consider the choreography in figure 6.3b on the preceding page as a service definition, then the participants Traveller and Hotel should always be considered a role unless explicit told otherwise. In SoaML it is done within the ServiceContract where the interface plays a role and this role is played by the participant in the ServiceArchitecture. In BPMN this distinction is not available in the diagrams. When modelling service oriented architectures,

modeling on a generic level with roles makes good sense; Why constrain the model to actual entities?

Consider another example that consists of three participants, e.g. a payment orchestration as modeled in figure 6.4 on the facing page. A payment orchestration could consist of a payer (the one who pays money), a payee (the one who receives money) and an acquirer (e.g. a bank or a credit card company). The conversation shows that the three roles communicate with each other and on what they are communicating about. The choreography shows in what order they communicate and who is initiating each communication. The orchestration provides us with detailed information on the interactions between the roles. These corresponding models use the role as the name of the participants. If we now should change the name into a concrete entity name, then the role each entity plays in the orchestration and choreography will be lost in the diagram. We cannot diagrammatically show both entity name and role name at the same time.

The modeling with roles uncover the lack of ability to define the distinction between a role and an entity in the BPMN diagram. This distinction is available in the SoaML version of the SOA. If we now choose to instantiate the roles in the BPMN Conversation diagram into some concrete entity, we would need to rename all participants in all the other diagram as well to understand the connections between them. In addition, considering the reuse of choreographies for other entities, this possibility would be lost when the choreography is made entity specific.

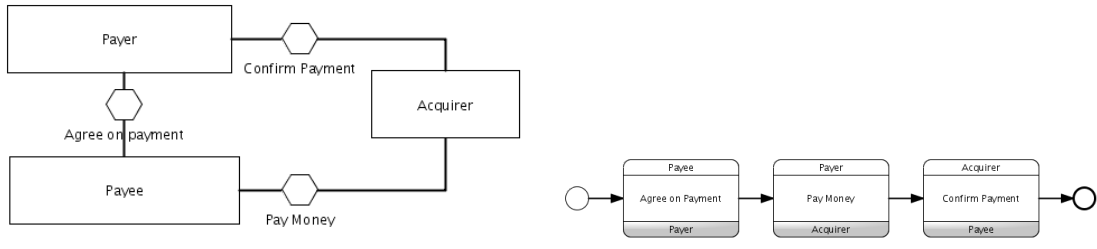
6.5 Mapping conclusions

By defining a modelling technique for BPMN 2.0, we can make the mapping against SoaML more direct. The ServiceContract and ChoreographyTask is basically the same element, with slight differences. The ServiceInterface pair could represent the same as the ChoreographyTask, but you need a pair to do it, and that makes it difficult to represent a participant with a service port. To solve this problem, we cut away ServiceInterface as an option for the mapping.

SoaML and BPMN 2.0 maps fairly good with some constraints on BPMN 2.0 modelling and use of the ServiceContract based approach for SoaML.

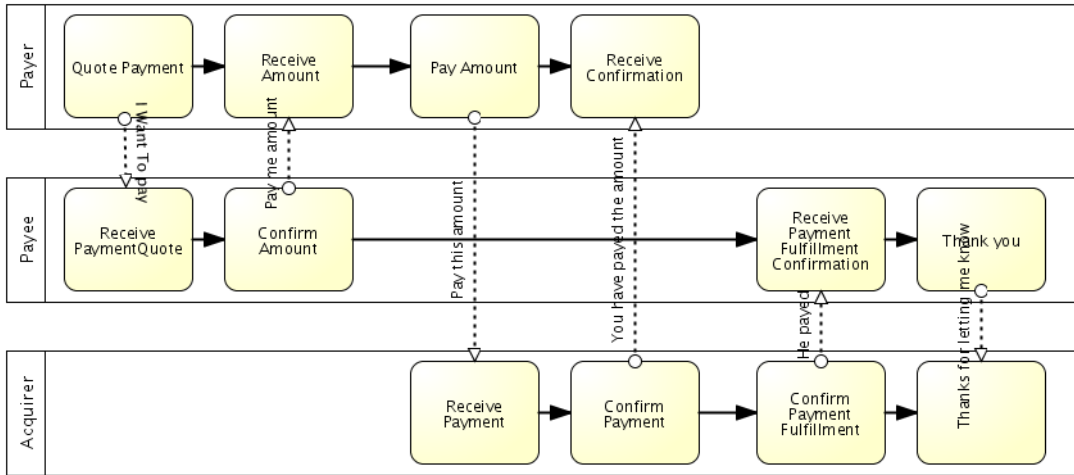
In the beginning of this chapter there are three approaches to the mapping, where the model-to-model technique was chosen. The evaluation provide the insight that this mapping in many cases cannot be done. Concepts seems similar, but they are different. BPMN4SOA's vision is to create a service oriented process modelling language. What needs to be done to make the vision for BPMN4SOA to come true is to use the strengths that SoaML provides and add them to BPMN 2.0. This actually is a combination of the other two approaches mentioned above, namely "none" and "meta model integration".

The previous chapter answered the research questions (as stated in section 1.2 on page 2) about strengths/weaknesses and capabilities of BPMN 2.0 and SoaML. This chapter identifies the overlapping concepts both in terms of modelling capability and how they can be combined into a new language.

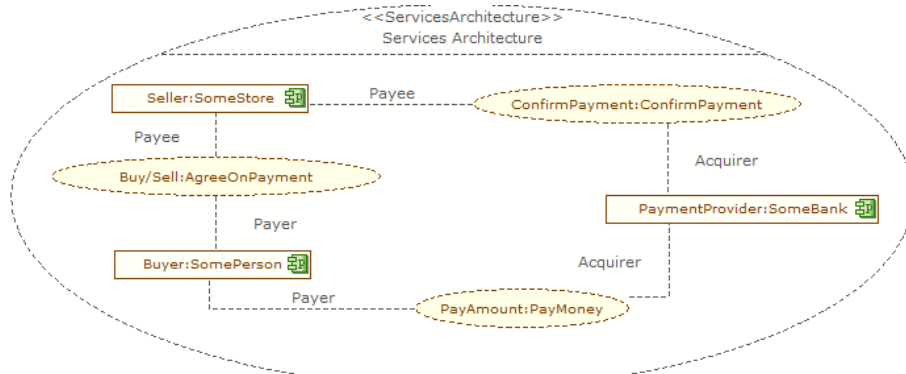


(a) Payment Conversation

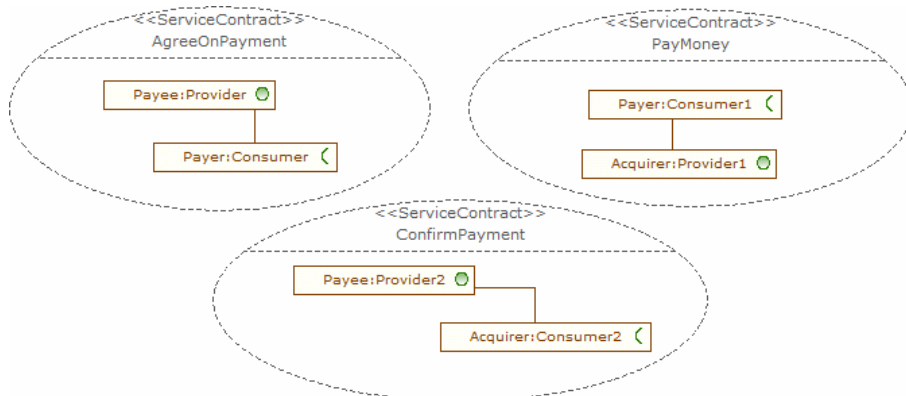
(b) Payment Choreography



(c) Payment Orchestration



(d) Payment Service Architecture



(e) Payment Service Contracts

Figure 6.4: A BPMN payment example consisting of three roles

The next chapter presents the specifications of BPMN4SOA, which accepts a separation of BPMN 2.0 and SoaML as separate paradigms, but adds extensions to BPMN 2.0 to realize SoaML capabilities in this process centric language.

The BPMN4SOA specification

This chapter presents the specification of the new language BPMN4SOA: A service oriented process modelling language.

7.1 Background

The evaluation of BPMN 2.0 and SoaML uncover that both languages do very well in the areas they are designed for and that they can represent the case as described in chapter 3 on page 11 in fairly similar ways. The mapping of the service architecture in SoaML against BPMN 2.0 reveal a way to use BPMN 2.0 to represent many of the same structures that SoaML can model. OOram provides hints to how BPMN4SOA can model roles in a better way.

Two paths have been presented: Add process capabilities to SoaML, or enhance service model capabilities and add information model capabilities to BPMN 2.0. For BPMN4SOA, the latter is chosen for some simple reasons: BPMN 2.0 is created as a completely new specification from the bottom up. BPMN 2.0 has extension capability that can be utilized, both for meta model extensions but also model references through Extensions and External Relationships.

BPMN4SOA is an extension of BPMN 2.0 to enable enhanced service modelling capabilities, constrains the use of MessageFlows and defines a method for information modelling intended for use in modelling the data sent by messages. BPMN4SOA should conform to Process Modeling Conformance and Choreography Modeling Conformance as defined in BPMN specification in a way that BPMN4SOA can be added to an already existing BPMN 2.0 implementation.

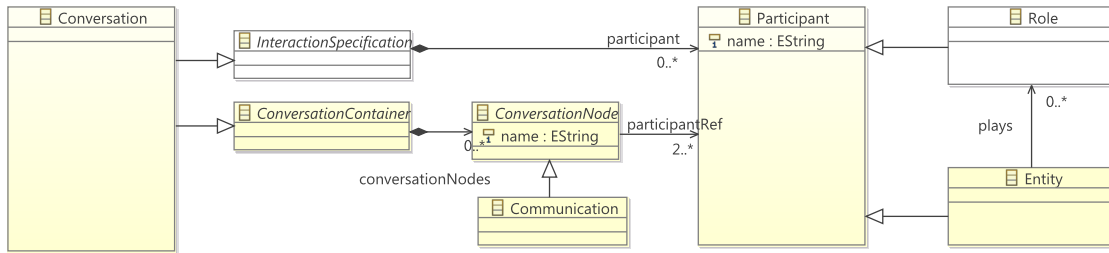


Figure 7.2: Conversation diagram meta model extended with Role construct

7.2 Role conversation diagram

Consider the meta model of the Conversation diagram in figure B.11 on page 108 and the meta model of Participant in figure B.6 on page 106. The figure shows that a Participant only can have one role.

“There is a many-to-many relationship between objects and roles: an object may play several different roles from the same or different roles models; and a roles may be played by several different objects” (25, p. 94).

When Reenskaug talks about objects he means concrete instances of a class in a system. For instance Per D. Ulyver from accounting can play the role as accountant, his profession, and the role as husband to his wife at the same time, but normally in different contexts. A participant should be considered as such an object instance and should therefore be able to play several roles in the same Conversation diagram. The case of Per cannot be modelled in a single Conversation diagram today. Thus, there needs to be multiple roles capability for a participant.

The BPMN 2.0 meta model Participant environment show that the Participant class has two possible attributes called PartnerEntity and PartnerRole. As a general basis, BPMN4SOA regards all participants as a Role. But in the Conversation diagram

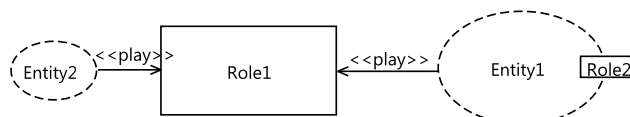


Figure 7.1: Roles displayed two ways

there is a need to be able to model both. Adding two sub classes of Participant, Entity and Role to the Conversation environment and adding the many-to-many relationship between these new classes, enables the diagram to model both the role abstraction and the concrete participant entity in the same diagram. The meta model in figure 7.2 is created as an Eclipse Ecore model and, thus, dont display an asterix on both ends of the “plays” association, as it would in UML.

- An Entity should be shown as an Ellipse with a dashed single line border.
- A Role should be shown as a Rectangle with a single line border

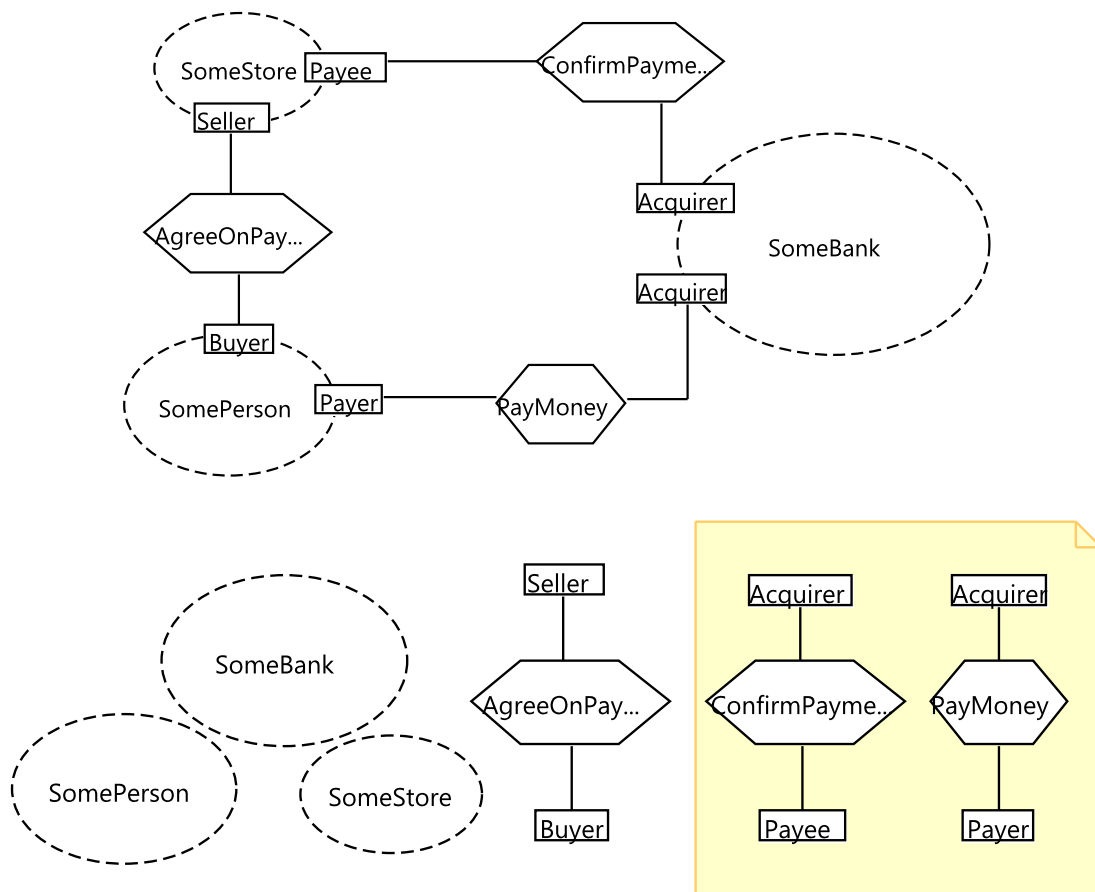


Figure 7.3: BPMN4SOA Role Conversation - upper shows the conversation. - lower shows an exploded version

When a participant plays a role, the role should be either associated with an arrow from the participant to the role, or it should be attached to the border of the participant who plays that role. Both examples can be seen in figure 7.1 on the facing page. The latter is used to represent a port through which a participant play a role in the communication. In some cases many entities can play the same role in the same diagram. For those cases the arrow representation is available to minimize lumping of the entities. Figure 7.3 upper part shows a BPMN4SOA Role Conversation model that corresponds to the Payment Conversation example in figure 6.3a on page 68.

When creating role models, we create abstract systems that we can hook into any interaction when we have the need for them. The role model is not bound by any entity, only position in some process. Figure 7.3 lower part show all the role models separated from any entities. The two role models inside the rectangular are the two models needed to fulfill a payment through a payment provider, commonly known as acquirer. A payer payes to the acquirer and the acquirer informs the payee about the transaction. This role model could be expanded into an escrow process. This modelling technique also ensures the separation of concerns for the conversation.

This extension could be added to BPMN by means of the Extensibility elements as discussed in section 5.1.2.1 on page 33.

7.3 Collaboration constraints

In BPMN 2.0 there are no rules for how MessageFlow should be grouped in a Choreography. It means that a ChoreographyTask can represent zero-2-many MessageFlows and there is no way to see in what extent a ChoreographyTask groups MessageFlow communication. There is a need for a specific intended granularity.

One ChoreographyTask should represent at most two message flows between two participants.

By adding this constraint, we define a concrete communication topic, with one operation/task initialization and a asynchronous response communication. The protocol is already implicate defined by the process, but the choreography diagram will show the sequence explicite. The first message sender is the initiator of the interaction.

Tasks that sends or receives messages should be defined as SendTask or ReceiveTask. The use of a SendEvent or ReceiveEvent is considered equal for the concern of the message flow, but tasks that performs something must be added. The use of undefined tasks can only be in private processes, and cannot send any messages.

7.4 BPMN4SOA information modelling

Information modelling is the ability to model concepts, relationships, attributes, operations and constraints. The extensions functionality that already exists in BPMN 2.0 (namely External Relationships as described in section 5.1.2.2 on page 34) is used to enable BPMN4SOA with information modelling. It could be argued that this could be done by defining a XML Schema editor to model, for instance, SOAP messages or other data centric XML schemas. However, if we want to use a model from a database definition designed in UML, we need to transform from PIM to PSM level. For BPMN4SOA, PIM level information models are considered a better option.

The external reference is done to UML. UML provides the entry level modelling capability compliance called Level 0 (L0) (18, p. 2). The level provides the capability to model class-based structures for the OO paradigm suitable for the modelling of the information data for the messages between participants.

By using the UML (Basic UML (18)) to model concepts, relationships and attributes, we can create the model for the data which is needed for operations in interfaces and the data exchanged in communications between participants of a process. By doing this, a model can be linked to any element of BPMN. (If we connected SoaML in this way, we could use the MessageType stereotype to model the data and link that model to BPMN.)

7.5 Conclusion

This chapter presents one extension to BPMN 2.0 to enable role modelling for BPMN4SOA. (The diagram notation is inspired by OOram role modelling view.) It also proposes a technique to constrain the use of choreography diagram together with collaboration limiting the number of message flows to a maximum of two per choreography thus defining a standard for a service contract. In addition BPMN 2.0 gets an external relation to UML compliance level 0 (L0).

BPMN4SOA tool implementation

Eclipse EMF/GMF is a framework for implementation of a graphical model editor. As proof of concept of the BPMN4SOA conversation diagram, this chapter provides the discussion on the work flow of implementing an editor in Eclipse GMF (Graphical Modelling Framework) based on a meta model represented in Eclipse EMF (Eclipse Modelling Framework project) and a discussion of the result of the implementation.

8.1 Eclipse EMF/GMF and Eclipse for SOA

“The sheer volume of useful modeling technologies that the Eclipse Modeling Project includes makes mastering a significant portion of it a daunting task. Even determining which specific available technologies are useful for solving any particular problem is a challenge exacerbated by the fact that, as a rule, the documentaton tends to lag far behind the development work” (14, p. xix) (sic).

The Eclipse for SOA (Service Oriented Architecture) project are developing a bundle comprising of several technologies for modeling, development and execution for the SOA paradigm. The package bundle has plugins for WSDL, Java webservices with Swordfish, SCA and other tools. Included in this project is a BPMN modeller. The editor is developed in the EMF/GMF framework but for the most part it is custom programmed and not generated code.

“Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle” (8). Eclipse is an extensible platform for software development. EMF and GMF are two of the modules that can be downloaded and used bundled with Eclipse.

EMF (Eclipse Modeling Framework) is a “Java framework and code generation facility for building tools and other applications based on a structured model” (9). The structured model can be annotated Java, Rational Rose (IBM), UML and more. EMF transforms your structured model into an .ecore representation of that model. Ecore is the native representation used by EMF. Ecore has its own metamodel and the package name is org.eclipse.emf.ecore.

GMF (Graphical Modelling Framework) “provides a generative component and runtime infrastructure for developing graphical editors based on EMF [...]” (9). GMF is an extension of GEF (Graphical Editing Framework) and add the ability to model components to generate a graphical model editor. GMF uses the .ecore model representation of EMF to generate a base for a graphical editor. By creating a set of graphical representations, GMF serves as the definer of the visible elements of the editor.

The BPMN editor was founded in 2006 by Intalio inc (10), and at that point, no meta model of BPMN 1.0 was provided by OMG. The meta model is therefore not consistent with any meta model for BPMN 1.0 or BPMN 1.2, but modified to suit the developer of the diagram editor. The lack of a meta model for the modeling language means that the tool vendors themselves need to create a meta model suitable for their implementation. Examples of this can be seen through out the meta model made for the Intalio BPMN editor. One of many examples of this is in the activity diagram element, eg. activity tasks and events where the Eclipse SOA BPMN diagram meta model uses one generic class `ActivityTask` with an attribute, which is an `Enumerator`, to define what type of activity it represents. This could be modeled using polymorphism as done in BPMN 2.0 meta model (10). This solution is probably chosen to make the programming of the system more programmer friendly and less complex. But must be taken into account when doing transformation on models both to and from the models created by this editor.

There exists an initiative for the implementation of BPMN 2.0 in Eclipse Modeling Development Tools (MDT) project. The implementation shall provide an open source reference implementation of the BPMN 2.0 specification (11). The status of the project is not known on basis of the information available on the Eclipse web pages but the repository shows some activity ¹.

¹http://dev.eclipse.org/viewcvs/index.cgi/org.eclipse.mdt/org.eclipse.bpmn2/plugins/?root=Modeling_Project

8.2 BPMN4SOA tool

Since there exists no good working implementation of BPMN 2.0 in Eclipse today, extending the whole implementation is too much work for this discussion. An implementation of BPMN4SOA is only a proof of concept, not a working example.

As a base for the editor, EMF need an Ecore model representation of the meta-model of BPMN4SOA conversation diagram. This is provided in figure 7.2 on page 74. The meta model is created from by combining the BPMN 2.0 meta model part in figure B.11 on page 108 and figure B.6 on page 106. To make the model simpler, the hierarchy of RootElement and BaseElement is skipped making the Conversation class the root element for the diagram.

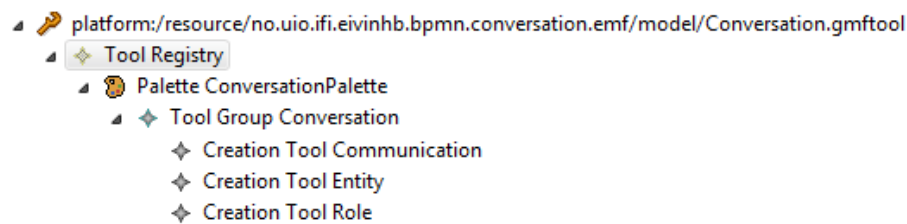


Figure 8.1: BPMN4SOA GMF Tool Registry model

GMF is a Model View Controller (MVC ²) based framework, where EMF provide the base for the model. The Model and the code for the view is generated by EMF by means of a .genmodel file. GMF generates the diagram controller by combining three models into a node mapping model: The meta model, the tool definitions and the graphical figure gallery.

The tool definitions are the definitions of creation tools. Figure 8.1 show the tree editor for the .gmftool file. The model consists of the three elements present in the diagram: Entity, Role and Communication. Normally the tool registry is generated quite well, but icons for the tools need to be changed.

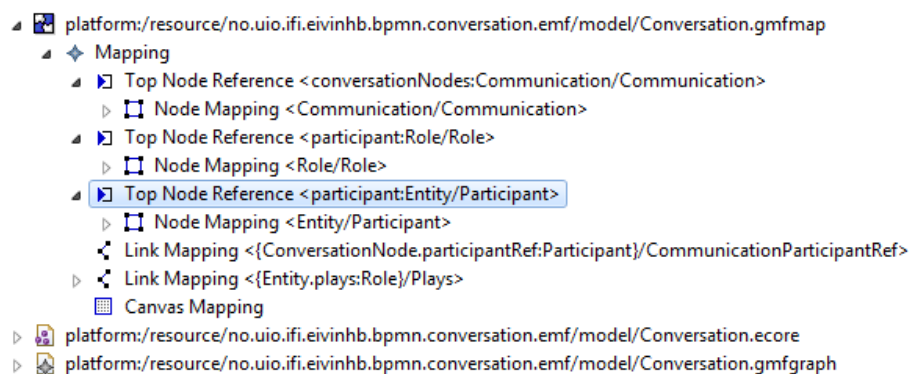


Figure 8.2: BPMN4SOA GMF Node Mapping model

To define the graphical look of diagram elements, the .gmfgraph file is generated. This file contains the node definitions, labels, figure descriptors, connections, compartments and the layout options for each elements. Basically, how everything looks

²<http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>

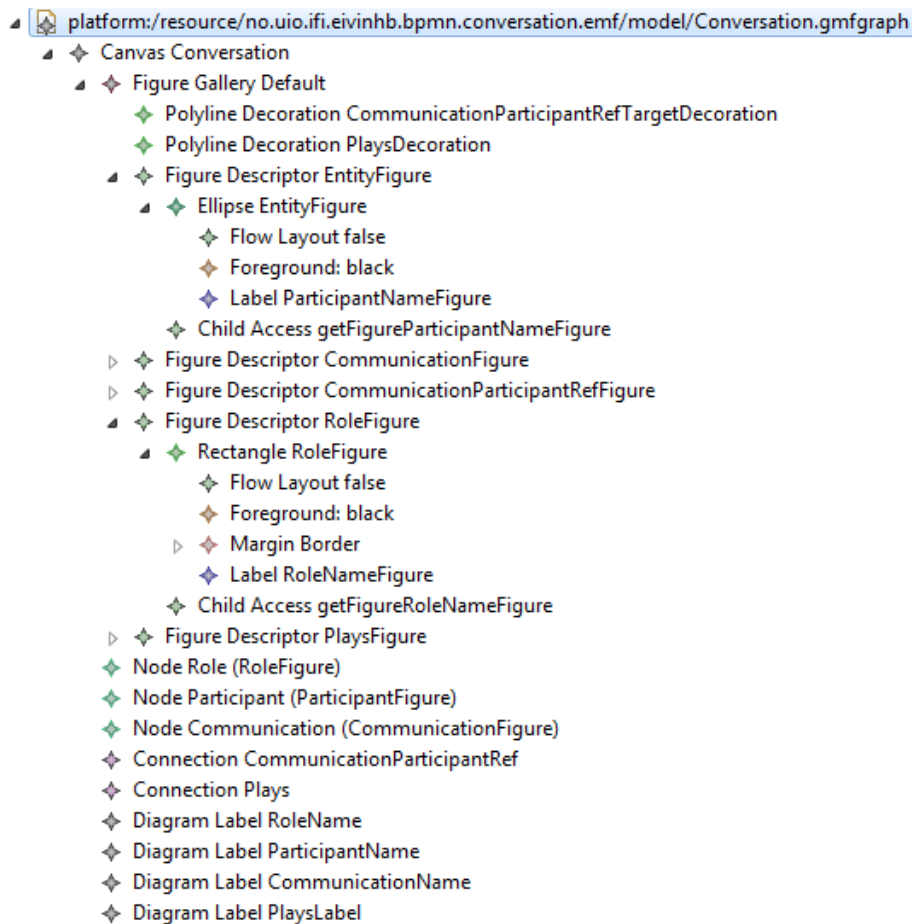


Figure 8.3: BPMN4SOA GMF Figure Gallery model

and contains other elements. GMF generated the main bulk of the graphical definitions, but normally the model will have to be created manually. Figure 8.3 shows that the Role figure is a black rectangular with a label for the name. The Entity is a black ellipse. The ellipse has a property to define the border as a dashed line. The Node element combine the figure and the diagram label into one node and the child access element makes it possible to edit the text label inside the figure.

The node mapping model, or `.gmfmap`, is the model that combine the creation tool, the graphical figure model and the meta model together. GMF generates the mapping model. But in practise, the model needs to be created manually. From the mapping model, GMF creates a diagram generator file and the GMF code generator creates the diagram editor.

Experience with GMF has shown that the generation model can only get you so far. GMF cannot layout a label to be centered in an element just by setting a layout manager. To do this, custom code needs to be implemented in the generated code. The excerpt of the generated `EntityEditPart.java` in listing 8.1 on the next page show the implementation of a custom layout manager to center the label in the ellipse. This simple example is a hint to why nearly all diagram code in the BPMN editor from Intalio is custom code, utilizing the GMF frame work but not the code generated in a great extent.

The models used of BPMN4SOA in all discussions on BPMN4SOA are made with

this proof of concept editor mainly generated from GMF models.

```

1  /**
2  * @generated NOT
3  * Custom layout to center label in the ellipse
4  * representing the Entity element.
5  * The getMinimumSize do not work properly for width
6  * of the label element. To solve the problem, we calculate an
7  * approximate width from the length of the string the label displays.
8  */
9  public EntityFigure() {
10     createContents();
11     this.setLayoutManager(new StackLayout() {
12     public void layout(IFigure figure) {
13         Rectangle r = figure.getClientArea();
14         List children = figure.getChildren();
15         IFigure child;
16         Dimension d;
17         for (int i = 0; i < children.size(); i++) {
18             child = (IFigure) children.get(i);
19             d = child.getMinimumSize(r.width, r.height);
20             d.width = Math.min((child.toString().length() * 6), r.width);
21             d.height = Math.min(d.height, r.height);
22             Rectangle childRect = new Rectangle(r.x
23                 + (r.width - d.width) / 2, r.y
24                 + (r.height - d.height) / 2, d.width, d.height);
25             child.setBounds(childRect);
26         }
27     }
28     });
29
30     this.setLineWidth(1);
31     this.setLineStyle(Graphics.LINE_DASH);
32     this.setForegroundColor(ColorConstants.black);
33 }

```

Listing 8.1: Custom layout manager for the Entity node

8.3 Usage and evaluation

The editor is a fairly simple GMF editor. The workspace is located on the left. This is where you create the diagrams. Each diagram consists of two XMI files where one is the model itself and the other is the diagram layout. The latter depends on the former. On the right is the tool list. By pointing in the empty space in the diagram, the create tool dialogue appears and present all the available nodes for creation as top nodes (ConversationNode, Role and Entity). Linking elements together is done by pointing on an element and a standard GMF link tool appear. The linking is done by dragging and dropping. If you select a link element, GMF provides appearance properties of the link where you can make the link look and feel different (Direct arrow, reclinear and smoothing of path). Properties of elements are also available. Figure 8.4 on the following page display most of these features.

This editor is only designed to create the BPMN4SOA role conversation diagram. Thus, this editor cannot be used to evaluate BPMN4SOA on the other extensions

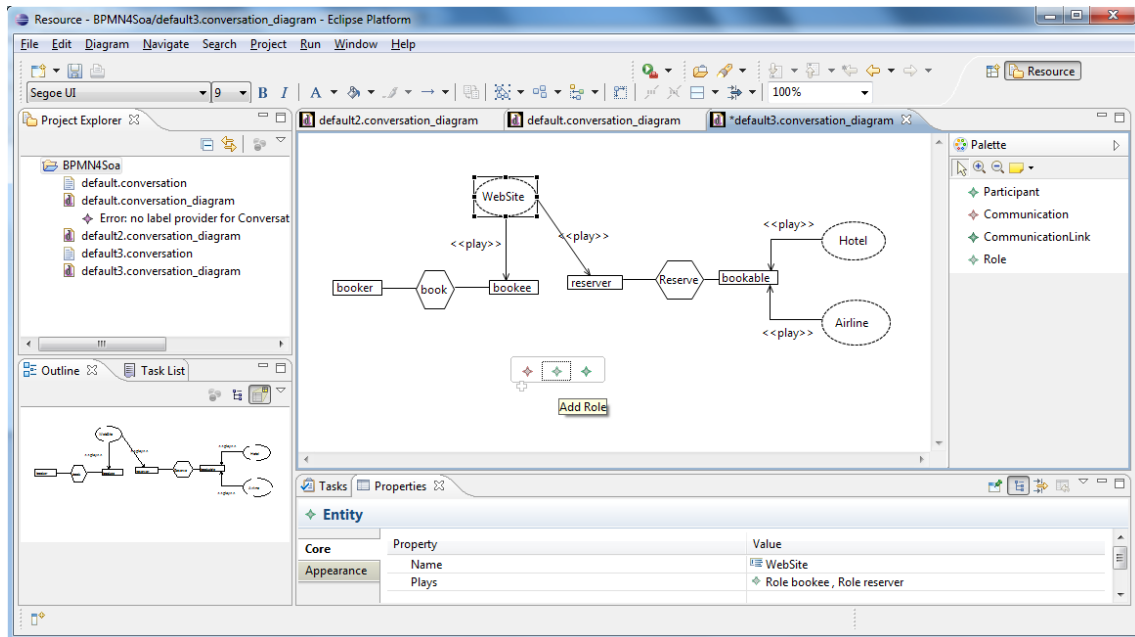


Figure 8.4: Screenshot of RoleConversation diagram editor

of BPMN 2.0 than the role concepts. However, the constraints and the information model external reference in BPMN4SOA can be evaluated without tool support. The evaluation of BPMN4SOA is provided in the next chapter, where the role concepts are evaluated based on the tool provided here.

Evaluation of BPMN4SOA

This chapter contains the evaluation of BPMN4SOA based on the requirements provided in chapter 4.3 on page 23. As with the evaluation of BPMN 2.0 and SoaML in chapter 5 on page 29, this evaluation of BPMN4SOA provides a case model with the available tooling.

9.1 BPMN4SOA travel case

The process model for the travel case in BPMN4SOA is the same models created in the BPMN 2.0 case evaluation. The constraints added for the collaboration and choreography in BPMN4SOA are already applied. The difference is that in the BPMN 2.0 evaluation, this was just a modelling technique. For the BPMN4SOA evaluation it is specified as a requirement. Figure 5.11 on page 40 and figure 5.12 on page 41 detail the process for the booking and the payment. It means that the lack of specific tooling for BPMN4SOA process modelling does not have any implication on the evaluation because BPMN4SOA can be modelled in the Signavio Saas tool.

Figure 9.1 on the next page shows the payment Role Conversation model of the travel case. There exists five entities: CreditCardCompany, Website, WebUser, Hotel and Airline. The web user plays three different roles as buyer, booker and payer. The web site plays the roles of booker, reserver, seller and payee. The credit card company (or the bank) play the role of acquirer and (payment) confirmer. Both Hotel and Airline play the role of bookable. The strengths of this approach is that all communications are generic through the roles, meaning that they can be reused for other participants or other processes. They define only how and what the communication is about, not binding the communication to whom in terms of a specific entity.

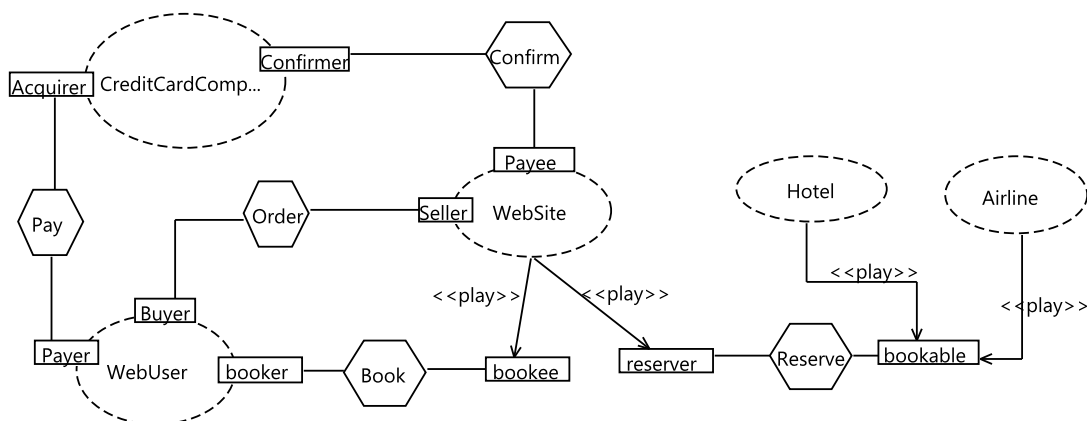


Figure 9.1: BPMN4SOA Role Conversation diagram for the travel case

The use of Choreography and Collaboration process is the same as in BPMN 2.0, but the communications are constrained to be done in the modelling technique as done in the case evaluation of BPMN 2.0. Additionally, the modeling of data for the messages are done with SoaML MessageType as depicted in the SoaML section for the case evaluation (Figure 5.18 on page 51).

9.2 Requirement evaluation

There are three focus areas of the requirements for a modelling language outlined in section 4.3 on page 23: Service, Information and Process modelling.

9.2.1 Service requirement evaluation

BPMN4SOA role conversation diagram provides extensions for role modelling and defines the use of participants as roles, unless stated otherwise.

Table 9.1: Scoring of requirements for service modelling in BPMN4SOA

Service modelling requirement	Score
Role and Entity distinction	5
Reusable service model elements	4
Service interaction protocol	4
Service provider/consumer	1
Initiator of a service	4
Interface definition	3
Composite structures	5

Role and Entity distinction is clearly defined and shown as two separate constructs in the BPMN4SOA Conversation diagram and meta model. All other

diagrams are now based on the fact that unless other stated, a participant is always a role.

Reusable service model elements BPMN4SOA provides the reusable Role Conversation construction that can be stored and reused for other processes and for other entities like a service contract, but the connection to the interface is still unclear.

Service interaction protocol. Same as BPMN 2.0

Service provider/consumer. Same as BPMN 2.0

Initiator of a service. Same as BPMN 2.0

Interface definition. Same as BPMN 2.0

Composite structures. Same as BPMN 2.0

9.2.2 Information requirement evaluation

BPMN4SOA uses the External Relationship feature of BPMN 2.0 to provide the information modelling capability. BPMN4SOA uses the UML at compliance level 0 for information modelling as described in SoaML specification for MessageType stereotype. Because of this, BPMN4SOA inherits the abilities of UML. The scoring is the same as for SoaML.

Table 9.2: Scoring of requirements for information modelling in BPMN4SOA

Information modelling requirement	Score
Definition of concepts	5
Definition of relationships	5
Definition of attributes	5
Definition of operations	5
Definition of constraints	5

The evaluation of this is done on a conceptual basis. The actual use of this is not tested. As seen (Meta model in figure B.5 on page 105), MessageFlow as a Message object has a property. This message could quite easily be defined as containing a object defined by a class concept hierarchy representing the data the message conveys.

9.2.3 Process requirement evaluation

BPMN4SOA specifies that all participants are considered as roles until specified otherwise. The result is a clearer definition of the choreography constructs, and makes them reusable, from Role Conversation diagram to Collaboration. The granularity of the concept role is easier to define than organisational units vs. system entity. For all other requirements, BPMN4SOA score the same as BPMN 2.0.

Table 9.3: Scoring of requirements for process modelling in BPMN4SOA

Process modelling requirement	Score
Participants and activities/tasks	5
Decision gateways	5
Event and error handling	5
Support for automated/human tasks	5
Model transformation and execution	5
Multi layer modelling	5
Public and private processes	5

9.3 Conclusions of evaluation

Since the class diagram is not implemented in a diagram editor, the actual use of BPMN4SOA for information modelling is not tested in a tool environment. But implementing this modelling capability in a BPMN 2.0 tool should be fairly straight forward and thus the testing could be performed. The proof of concept tool for BPMN4SOA shows the new capabilities added to the Role Conversation diagram and outline the possibilities for reusable contracts and consequently collaborations. The process modelling of BPMN4SOA can be realized with use of the Signavio SaaS tool.

Conclusion and future work

This thesis has evaluated the soon to be complete specifications of BPMN 2.0 and SoaML, and used both languages to model the same case description. Section 1.2 on page 2 asks some research questions and the evaluation has uncovered strengths and weaknesses for both languages, and similarities and differences they may have. Experience from the case modelling provides insight for the mapping of similar notation and capabilities to each other. The mapping uncovered a way to create a new language that is called BPMN4SOA. This language builds on BPMN 2.0, extending it with a concept of role and through reference to UML give BPMN4SOA information modelling capabilities. By defining a technique for modelling messages between roles, BPMN4SOA now has a better definition for the use of the concept of participant for collaboration modelling and the BPMN4SOA specification provides information modelling capabilities.

10.1 Conclusion of requirements evaluation

SoaML, BPMN 2.0 and my proposal BPMN4SOA are all evaluated on a set of requirements. Table 10.1 on the following page contains the results of the various evaluations.

From the table we can see that SoaML score best in its service modelling requirements. A general conclusion for this is that SoaML is specifically created for this use. BPMN4SOA builds on BPMN 2.0 but extends it in terms of role modelling and consequently scores better in that requirement. BPMN4SOA does not address the lack of service provider/consumer specification, which is the biggest weakness BPMN4SOA has compared to SoaML. BPMN4SOA provides better interface modelling than BPMN 2.0 due to the coupling with UML for information modelling.

BPMN4SOA scores better in the reuse of service elements, due to a definition of the role concept as a standard for participant. BPMN4SOA inherits the information modelling from UML, and thus gets the same score as SoaML. BPMN4SOA also inherits the process modelling abilities that BPMN 2.0 provides, but scores better in the multi layer modelling due to the constraint added to the participant construct, in effect making it a reusable role.

Table 10.1: Comparing requirement scores for BPMN 2.0, SoaML and BPMN4SOA

Requirement	BPMN 2.0	SoaML	BPMN4SOA
Service model			
Role and Entity distinction	2	4	5
Reusable service model elements	3	5	4
Service interaction protocol	4	4	4
Service provider/consumer	1	5	1
Initiator of a service	4	3	4
Interface definition	3	5	4
Composite structures	5	5	5
SUM	22	31	27
Information model			
Definition of concepts	2	5	5
Definition of relationships	1	5	5
Definition of attributes	1	5	5
Definition of operations	3	5	5
Definition of constraints	1	5	5
SUM	8	25	25
Process model			
Participants and activities/tasks	5	4	5
Decision gateways	5	3	5
Event and error handling	5	4	5
Support for automated/human tasks	5	2	5
Model transformation and execution	5	4	5
Multi layer modelling	4	3	5
Public and private processes	5	3	5
SUM	34	23	35
SUM overall	64	79	87

Figure 10.1 on the next page displays the Zachman framework, now with BPMN4SOA added to the picture. BPMN4SOA cover BPMN 2.0 completely and BPMN4SOA has some abilities in the data column thus extending the coverage compared with BPMN 2.0. This looks a bit strange due to the offset of the blob for BPMN4SOA. This is due to the fact that information modelling capability added to BPMN4SOA is not part of the enterprise model, but only what UML adds through L0 compliance.

There are many aspects of BPMN 2.0 and SoaML that have not been taken into account. Examples in BPMN 2.0 are the multiplicity of participants in all diagram types, callable elements, discussion on the human/manual tasks vs the computer

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>
Objective/Scope (contextual) <i>Role: Planner</i>	List of things important in the business	List of Business Processes	List of Business Locations	List of Important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) <i>Role: Owner</i>	Conceptual Data/ Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) <i>Role: Designer</i>	Logical Data Model	System Architecture	Contributed Items Architecture	Human Interface Architecture	Processing	Business Rule Model
Technology Model (physical) <i>Role: Builder</i>	Physical Data/Class Model	Technology Design Model SoaML	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Representation (out of context) <i>Role: Programmer</i>	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Speculation
Functioning Enterprise <i>Role: User</i>	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

Figure 10.1: Zachman framework coverage for BPMN4SOA

driven tasks of Send/Receive/Service/Script-tasks etc., all of which are capabilities that BPMN4SOA inherits. The modelling technique and methodology on these constructs are not a part of this discussion. It has to do with the fact that tooling is not implemented for process execution environments, thus, concrete cases are difficult to evaluate. For the discussion in this thesis, they are identified but not evaluated to the full extent. Another issue is the actual multi diagram modelling and linking of these different abstraction levels. They have been evaluated on a conceptual level, but not concrete level. Thus, the conclusions could be different if a proper tool environment could be used.

For SoaML, the use of activity diagrams for protocol modelling is not done. This is because the SoaML specification doesn't cover all aspects of how this should be done. Consequently, there are still questions on how to utilize all aspects of SoaML.

10.2 Future work

As the requirement evaluation shows, the BPMN4SOA is not as good as it should be. The requirements are not met to the full extent. This is especially apparent in the provide/consumer requirement and the initiator of a service. Following is the next iteration (that I would perform if time allowed it). This is only a conceptual suggestion, not a solution and it is not implemented in any tools (models are drawn).

Figure 10.2 on the following page displays the role class with two specialisations inheriting the role: Consumer and Provider. By adding these two sub classes, we could enable the modelling language to distinguish between the provider and consumer. This could be done while still being compliant with BPMN 2.0 in the same way the role is a subclass of Participant class. This enables two types of available roles in the tool. The set of two roles with a conversationNode are associated with a ChoreographyTask that specifies the initiator of the interaction. Now, there are three extra information features that could be added to the role conversation diagram.

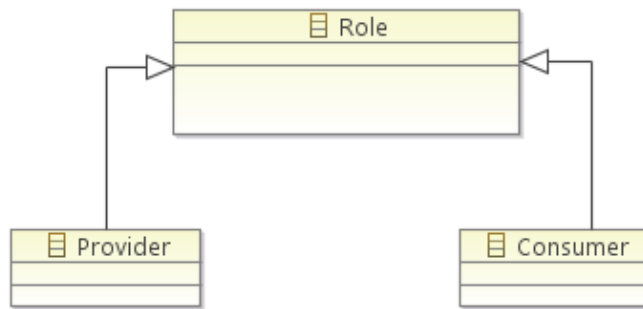


Figure 10.2: Specialisation of the role concept for future work

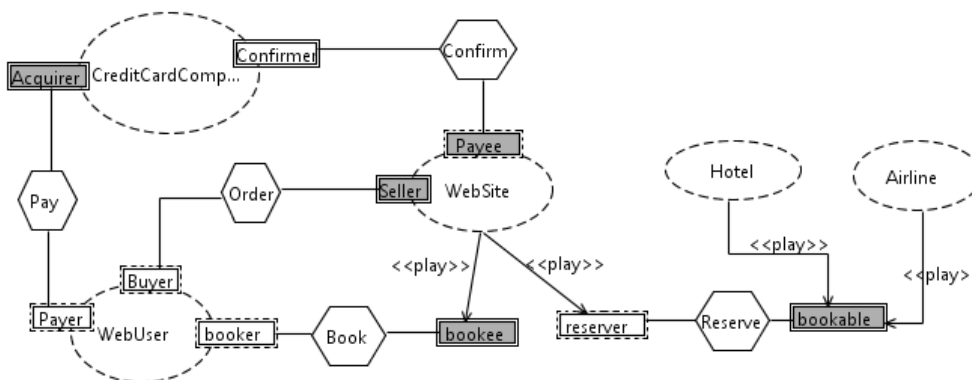


Figure 10.3: Example drawing of the the extensions to the role conversation diagram for future work.

Figure 10.3 shows a modification of the case model in figure 9.1 on page 86. Changes are drawn in. Each communication has a grayed out role. The one not grayed out is the initiating part. We can also see that each conversation has a role with an extra solid border and one with an extra dashed border. The solid border represent the provider of a service and the dashed represent the consumer of the service. The idea is based on OOram, but should not be confused with the multiplicity that OOram provides, with border on communications.

We now have enabled in the role conversation diagram should fulfill the requirements for provider/consumer and initiator of service. This could possibly raise the scores for BPMN4SOA service modelling to a sum of 33, bringing the total sum for BPMN4SOA from 87 to 93.

We still do not have any interface definition capabilities in the diagram for BMPN4SOA. This could probably be done in the UML editing part of a modelling tool, but since BPMN 2.0 provide its own interface constructs, they should probably not be combined. OOram provide the Interface view that could be used as inspiration for how it could be implemented.

Another interesting topic to explore could be information modelling for messages by XML schemas, not UML. There are pros and cons for this approach: The realisation would be a PSM model, not a PIM model. Thus, the use of XML schemas for information modelling limits the reusability of the model.

References

- [1] Jim Amsden. Bpmn and soaml integration. http://www.omgwiki.org/bpmn2.0-fff/doku.php?id=public:sub-teams:soaml_coordination, nov 2009.
- [2] Arne-Jørgen Berre, Brian Elvesæter. Model based system development, part 1: Mde - model driven engineering. <http://www.uio.no/studier/emner/matnat/ifi/INF5120/v08/undervisningsmateriale/INF5120-Part-I-MDE.pdf>, 2008.
- [3] ATHENA. Advances technologies for interoperability of heterogeneous enterprise networks and their application. athena a2.2, 2005. DFKI.
- [4] Bill Karakostas, Yannis Zorgios. *Engineering Service Oriented Systems - A Model driven Approach*, volume 1. IGI Publishing, 2008.
- [5] Brian Elvesæter, Arne-Jørgen Berre. Omg specifications for enterprise interoperability. Standards Workshop at I-ESA 2010, April 13th 2010, Coventry, UK.
- [6] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, 1989.
- [7] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks. *EMF Eclipse Modeling Framework*, volume 1 of 2. Addison-Wesley, 2009.
- [8] Eclipse. About the eclipse foundation. <http://www.eclipse.org/org/>, October 2009.
- [9] Eclipse. Graphical modelling framework. <http://www.eclipse.org/modeling/gmf/>, oct 2009.
- [10] Eclipse. Eclipse soa. <http://www.eclipse.org/stp/>, jan 2010.
- [11] Eclipse. Mdt/bpmn2. <http://www.eclipse.org/modeling/mdt/>, April 2010.
- [12] Davor Gornik. Entity relationship modeling with uml. http://www.ibm.com/developerworks/rational/library/content/03July/2500/2785/2785_uml.pdf, nov 2003. [Online; accessed 19-April-2010].

- [13] Ulrike Greiner, Sonia Lippe, Timo Kahl, Jörg Ziemann, and Frank-Walter Jäkel. A multi-level modeling framework for designing and implementing cross-organizational business processes. In *Technologies for Collaborative Business Process Management*, pages 13–23, 2006.
- [14] Richard C. Gronback. *eclipse Modeling Project. A Domain-Specific Language (DSL) Toolkit*, volume 1 of 1. Addison-Wesley, 2009.
- [15] Ryan K. L. Ko. A computer scientist’s introductory guide to business process management (bpm). *Crossroads*, 15(4):11–18, 2009.
- [16] Michael L. Brodie, Joachim W. Schmidt, John Mylopoulos. *On Conceptual Modelling*, volume 1. Springer-Verlag New York inc, 1984.
- [17] O. Nikiforova, A. Cernickins, and N. Pavlova. Discussing the difference between model driven architecture and model driven development in the context of supporting tools. In *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, pages 446–451, Sept. 2009.
- [18] OMG. Omg unified modeling language (omg uml 2.1.2), infrastructure. <http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/>, nov 2007. [Online; accessed 11-Januar-2010].
- [19] OMG. Omg unified modeling language (omg uml 2.1.2), superstructure. <http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/>, nov 2007. [Online; accessed 11-Januar-2010].
- [20] OMG. Business process model and notation (bpmn) ftf beta 1 for version 2.0, aug 2009. Object Management Group (OMG).
- [21] OMG. Business process model and notation (bpmn) v0.9.15, March 2009. Object Management Group (OMG).
- [22] OMG. Service oriented architecture modeling language (soaml) specification, dec 2009. Object Management Group (OMG).
- [23] Chun Ouyang, Marlon Dumas, Wil M. P. Van Der Aalst, Arthur H. M. Ter Hofstede, and Jan Mendling. From business process models to process-oriented software systems. *ACM Trans. Softw. Eng. Methodol.*, 19(1):1–37, 2009.
- [24] Philip Kotler, Gary Armstrong, John Saunders, Veronica Wong. *Principles of marketing*, volume 3. Pearson Education Limited, 2001.
- [25] Trygve Reenskaug. Working with objects - the ooram software engineering method. <http://heim.ifi.uio.no/~trygver/1996/book/WorkingWithObjects.pdf>, mar 1995. [Online; accessed 19-December-2009].
- [26] SHAPE. Purchase order reference example, April 2008. Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering (SHAPE).
- [27] Bruce Silver. *BPMN Method and Style*, volume 1. Cody-Cassidy Press, 2009.
- [28] Richard Soley. Model driven architecture. <http://www.omg.org/cgi-bin/doc?omg/00-11-05>, nov 2000. Object Management Group (OMG).
- [29] Wikipedia. Integrated enterprise modeling — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Integrated_Enterprise_Modeling&oldid=326068290, 2009. [Online; accessed 12-January-2010].

- [30] Wikipedia. Model-driven engineering — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Model-driven_engineering&oldid=327857069, 2009. [Online; accessed 27-November-2009].
- [31] Wikipedia. Event-driven process chain — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Event-driven_process_chain&oldid=337973276, 2010. [Online; accessed 25-February-2010].
- [32] J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292, 1987.
- [33] John A. Zachman. John Zachman’s concise definition of the the zachman framework. <http://www.zachmaninternational.com/concise%20definition.pdf>, 2010. [Online; accessed 21-February-2010].

Appendices

OOram - Role Modeling

The OOram methodology is not the newest on the market being at least 15 years old. But in terms of Role Modeling it still has some modeling constructs worth taking into consideration. Some of the diagrams used in OOram has similar implementations in UML, some of which will be presented here. The main focus in this discussion is the concept of Role Modeling and digrams created for use in the OOram methodology. The idea behind OOram was to create a candidate for future modeling languages combining the strengths of object and class. The class is the abstract description of an object.

“The class/object duality is as essential to object oriented programming as it is disruptive to object oriented modeling. A future modeling standard should be built on a unified conceptual framework with sufficient expressive power to describe all interesting aspects of an object system within a single, integrated model” (25, p. 2).

The OOram role model does just that. Reenskaug argues that all information that can be expressed either by a class based model or a object model can both be expressed by the same role model. He saw the synergy of this merge of the concepts of class and object to increase the leverage of decomposition of large systems and for reuse of already identified and proven components. “In an OOram model, patterns of interacting objects are abstracted into a corresponding pattern of interacting roles” (25).

A Role is an abstraction belonging to the realm of modeling, Class belongs to the realm of implementation, he argues. An object can play different roles in interaction with other objects. The object belongs to the same class if it has the same properties regardless of role it plays. An example from OOram is the role of a dining room and

the role of the kitchen. Both roles can be expressed by the same class, a room, and can have the same properties. Role modeling provides the separation of concern by describing different phenomena in different role models.

A basic assumption in OOram is the relationship between type, object, role and class and the fact that there is a many-to-many relationship between them. (As stated in (25))

1. The object is the "is" abstraction and represents a part of a system. An object has identity and attributes, and it is encapsulated so that the messages it sends and receives constitute all its externally observable properties.
2. The role is the "why" abstraction. All objects that serve the same purpose in a structure of collaborating objects as viewed in the context of some area of concern are said to play the same role.
3. The type is the "what" abstraction. All objects that exhibit the same externally observable properties are considered to belong to the same type.
4. The class is the "how" abstraction. All objects that share a common implementation are considered to belong to the same class.

The object oriented paradigm and the role paradigm is covered in the introductory subsections 3.4.1 on page 14 and 3.4.2 on page 15. Lets consider the graphical notation of OOram role modeling. Reenskaug proposed an easy to understand notation based on simple constructions. OOram consists of several different views, namely: Area of concern view, Stimulus-response view, Role List view, Semantic view, Collaboration view, Scenario view, Interface view, Process view, State Diagram view and the Method Specification view. Only the Collaboration, Scenario and Interface view is in scope of this discussion.

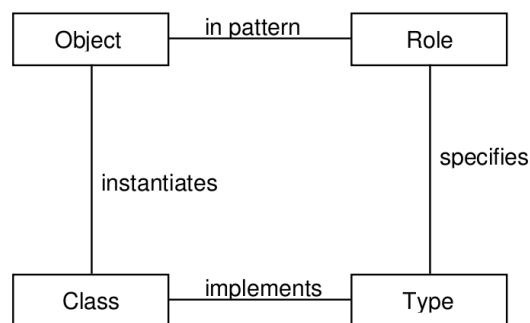


Figure A.1: Many-to-many relationship between type, object, role and class.

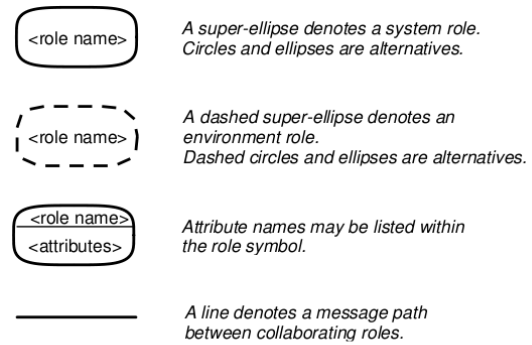
A.1 Collaboration view

The Collaboration view is the diagram that shows the roles and the message paths between the roles, i.e. how the roles interact with each other.

We see the role is represented by a super-ellipse with a solid border and is called a system role. The ellipse with a dashed border is an environment role meaning that the role is the initiator or trigger of the activities in the system. The lines represents the message paths between the collaborating roles. The cross say that the adjacent

role does not know about the role on the other side while the simple circle denotes a port and that the adjacent roles knows about exactly one collaborator. The double bordered circle represents multiple collaborating roles known to the adjacent role one of which is displayed in the view.

The choice of a super-ellipse is down to recommendations by James Martin ¹ saying that data should be represented by square-cornered rectangles and activities would be represented by a rounded rectangle. Reenskaug chose the super-ellipse since it is a middle way. A normal ellipse is also accepted in OOram (25, p. 98).



Let's consider an example of a OOram collaboration view. In the figure A.3 we see four roles with message paths between them. The roles are Vendor, Enterprise, PayeeBank and PayerBank. The Enterprise is an environment role because this role sends a stimulus message; the enterprise wants something. The Vendor is an environment role because it receives a final message; supply goods. One aspect to have in mind is that the role played as the payer bank and the payee bank actually could be the same object, or participant, playing different roles. The separation of concern the role model provides lets us look away from that fact and only focus on the actual interactions taking place.

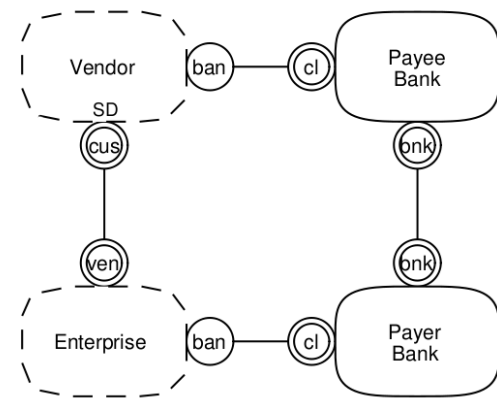


Figure A.3: Collaboration view example

The ports on the message paths also tells us something interesting. Focusing the message from Enterprise to PayerBank we see that the Enterprise knows only of one bank, but the bank knows about many customers, one of them is represented in this view, namely the Enterprise. Looking at the message path between the Enterprise and the Vendor, we see that both know about many vendors and customers accordingly, but again only one of which is in this view. An interesting aspect of this diagram view is that if we erase all other roles except the Enterprise, we can still see who and how many the Enterprise is communicating with via the ports on the message path.

OOram also provides a variation of the system role called the virtual role. The virtual role is an aggregation of same objects. This could be used on the example above to lump together the payee and payer bank into a single bank virtual role. This construct is only for convenience for the modeler and does nothing to the actual role models. The virtual role is represented by a solid bordered super-ellipse with a shadow behind.

¹James Martin: Recommended Diagramming Standards for Analysts and Programmers: A Basis for Automation. Prentice-Hall, Englewood Cliffs, NJ 1987 (ISBN 0- 13-767377-9 025)

A.2 Interface view

The interface view is basically an addition to the collaboration view displaying the specification of interfaces used at different ports. The interface is not an implementation of an interface as we do in Java, for instance, but can be thought of as the same abstraction as capabilities are in SoaML.

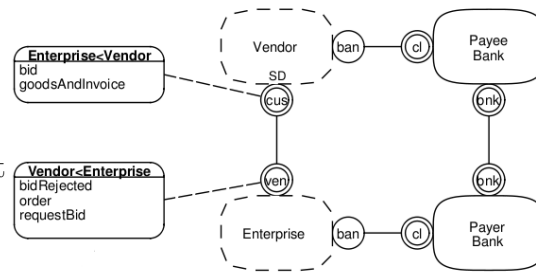


Figure A.4: Interface view example

Figure A.4 shows the interface is a super-ellipse with two parts, one for the name and one for a list of messages. The messages are descriptive of what capabilities the role has through that particular port. The naming convention of the interface in this example ² are all the messages that can be sent to Vendor from Enterprise. This is opposite of what are used in, for example, Java where we request an interface based on an operation and gets an answer. In OOram, the response is specified in the opposite interface.

A.3 Scenario view

“A Scenario is a description of a specific, time ordered sequence of exemplifies a interactions between objects. An interaction represents the event of transmitting a message from a sender object to a receiver object” (25, p. 100-101).

In the scenario view of OOram (similar to the UML sequence diagram) the strict sequence of the messages in the interaction between roles are shown as arrows between the roles in a strict time line. Consider the first communication in figure A.5 from Enterprise to Vendor through the message “requestBid”. The message name is a part of the Vendor<Enterprise interface. Vendor replies with a message going the other way called “bid” which is a part of the Enterprise<Vendor interface. In UML sequence diagram, this would be the other way around. Enterprise would use the operations that Vendor is supplying. Enterprise is an environment role; It initiates the interaction. Further more in the scenario view, the last message is the one going from PayeeBank to Vendor that the Enterprise has payed the money. Vendor is the final interacting role in this role model and is an environment role.

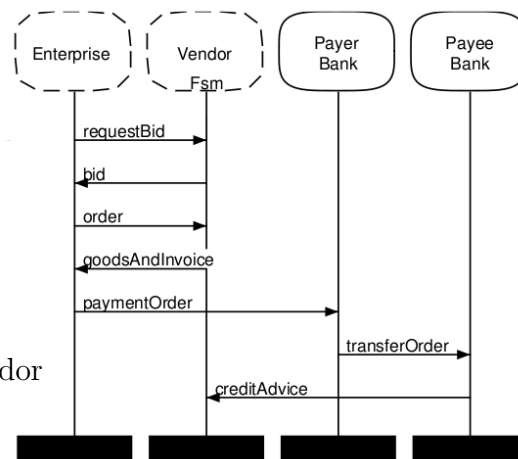


Figure A.5: Scenario view example

²Vendor<Enterprise meaning Vendor-from-Enterprise

Selected BPMN 2.0 meta models

This appendix chapter contains a selection of meta models from the BPMN 2.0 specification used for reference on discussion about BPMN 2.0. The models are all copied from BPMN 2.0 specification.

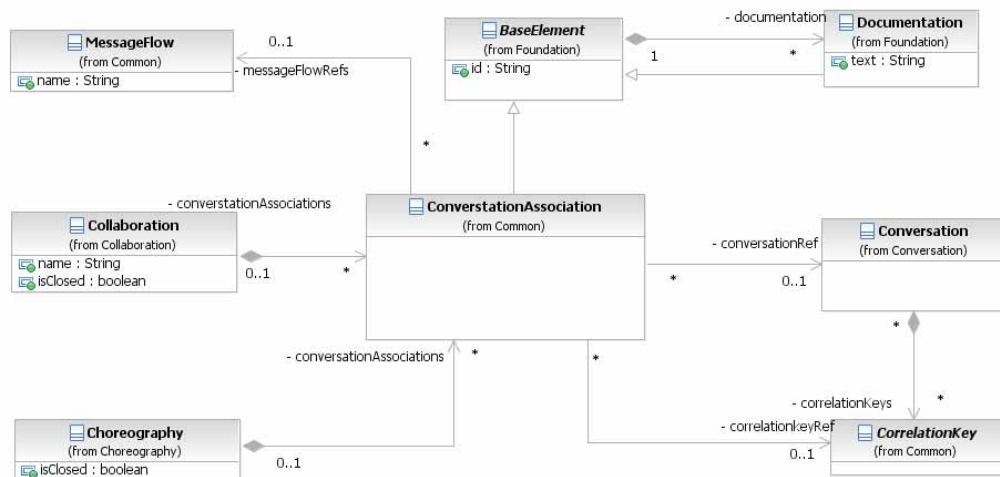


Figure B.1: BPMN 2.0 ConversationAssociation class diagram

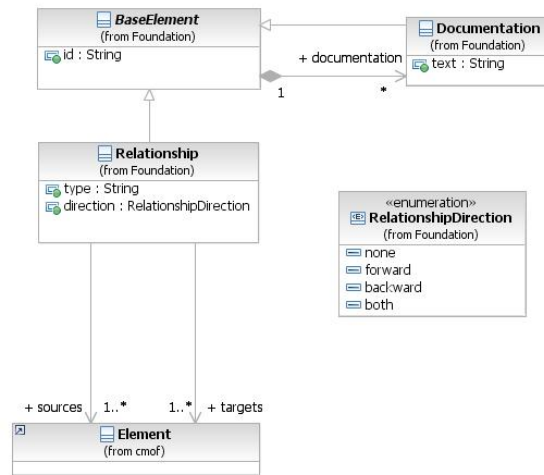


Figure B.2: BPMN 2.0 External Relationships

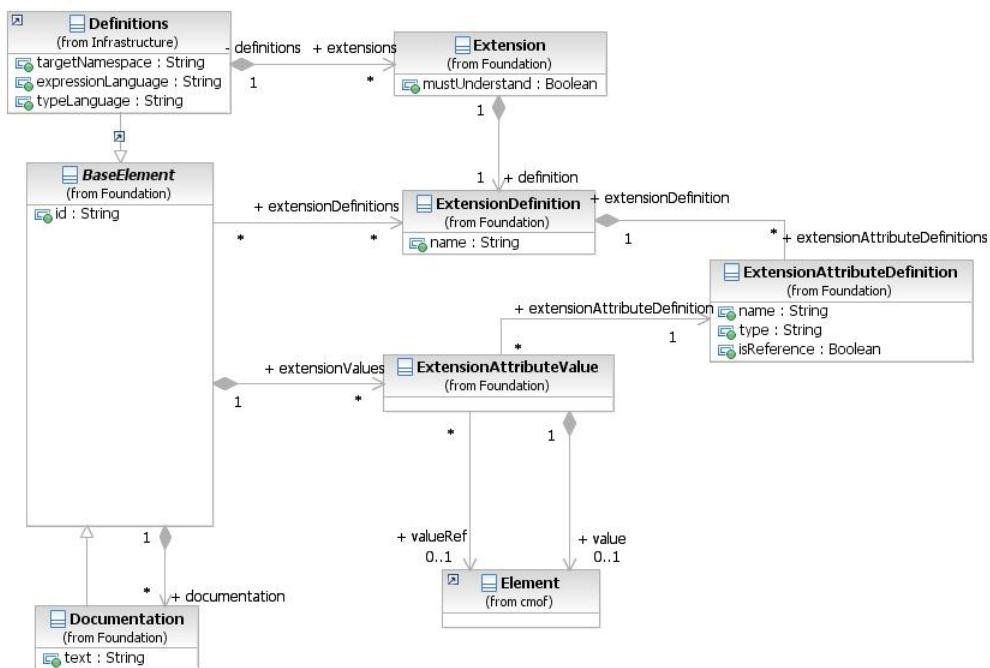


Figure B.3: BPMN 2.0 Extensions class diagram

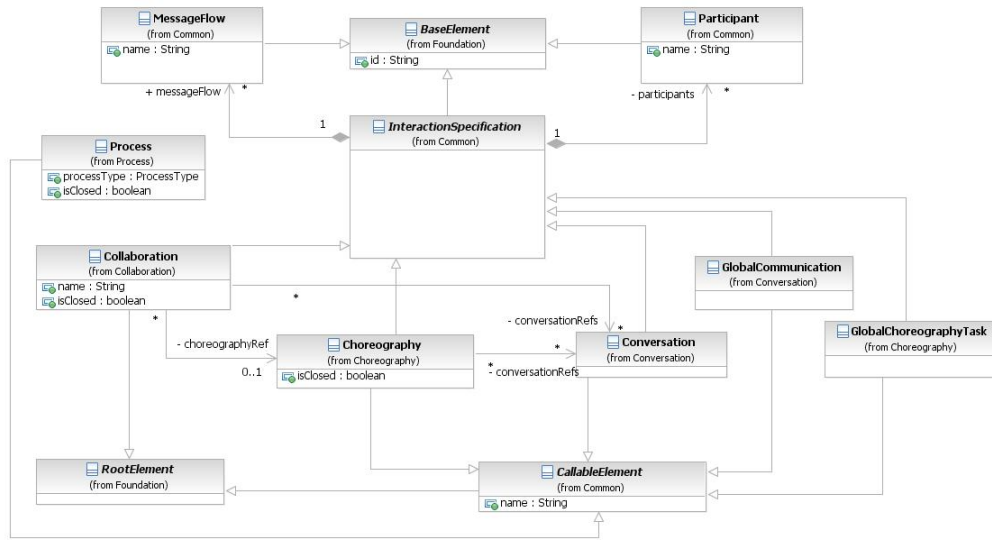


Figure B.4: BPMN 2.0 Interaction Specification class diagram

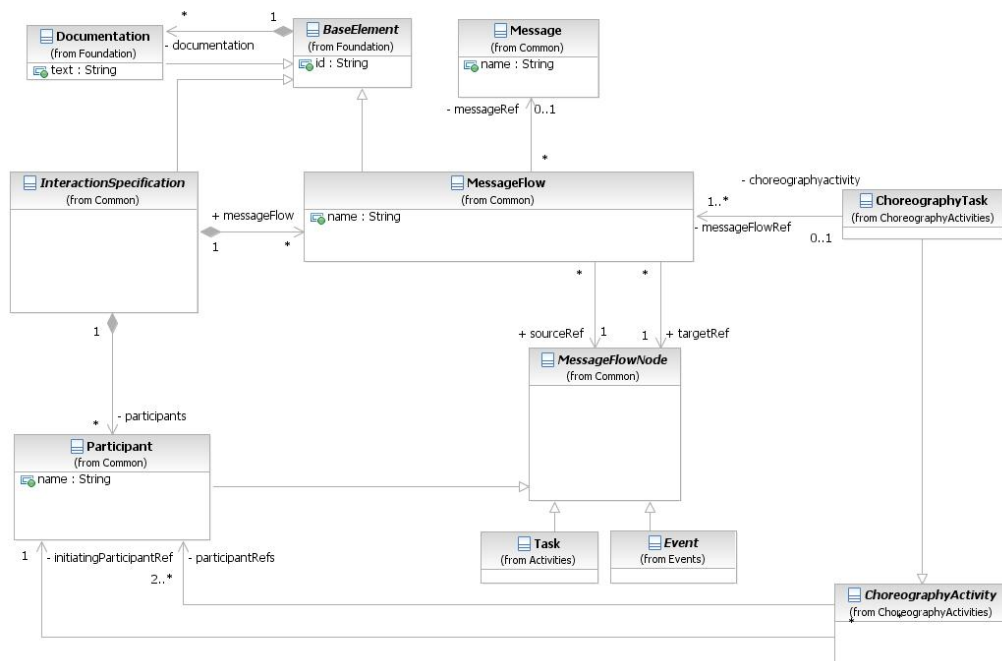


Figure B.5: BPMN 2.0 Message flow class diagram

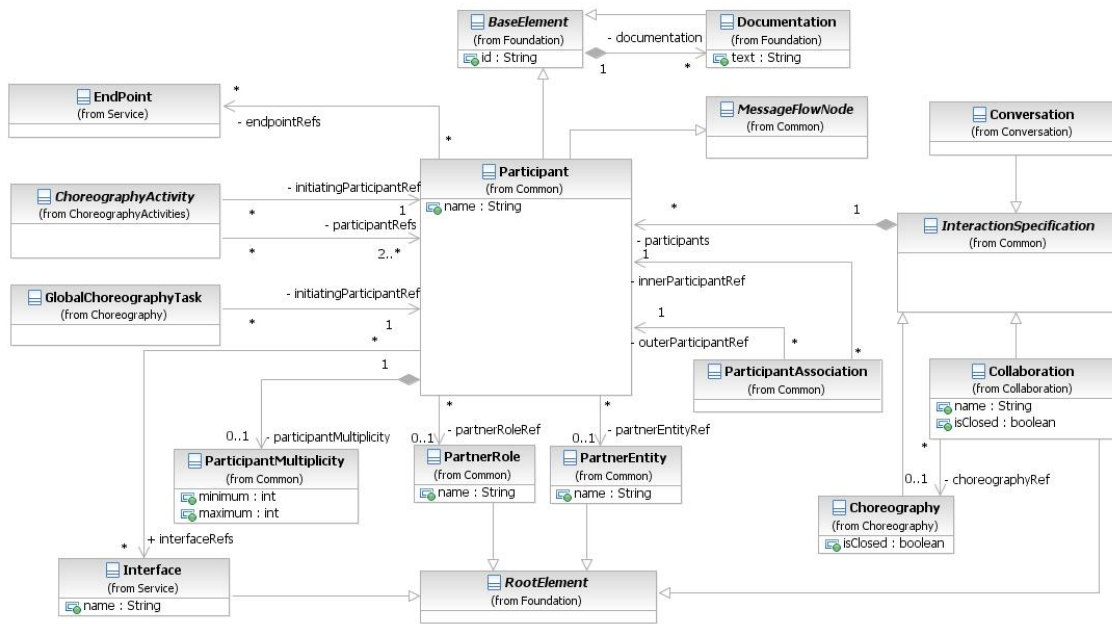


Figure B.6: BPMN 2.0 Participant class diagram

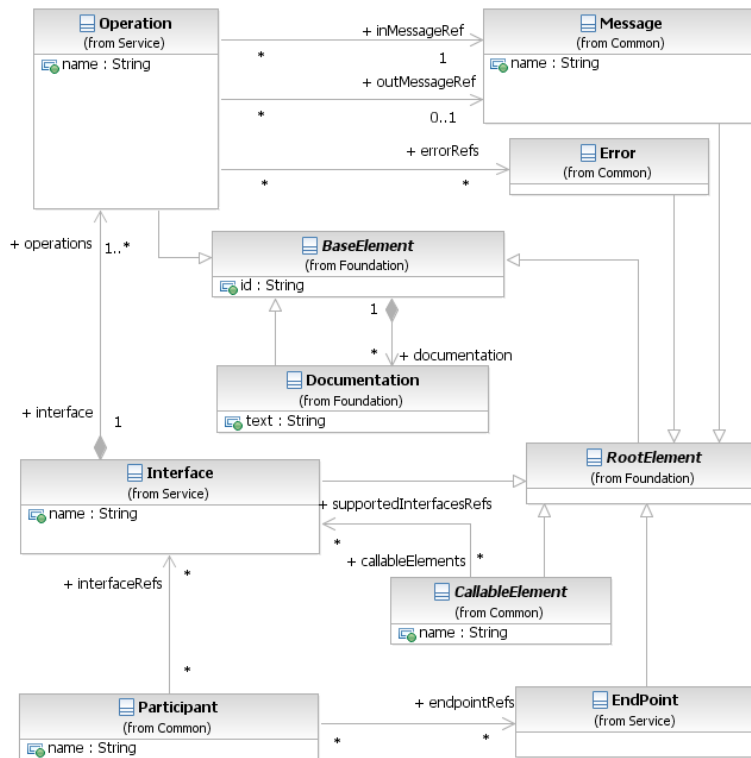


Figure B.7: BPMN 2.0 Services class diagram

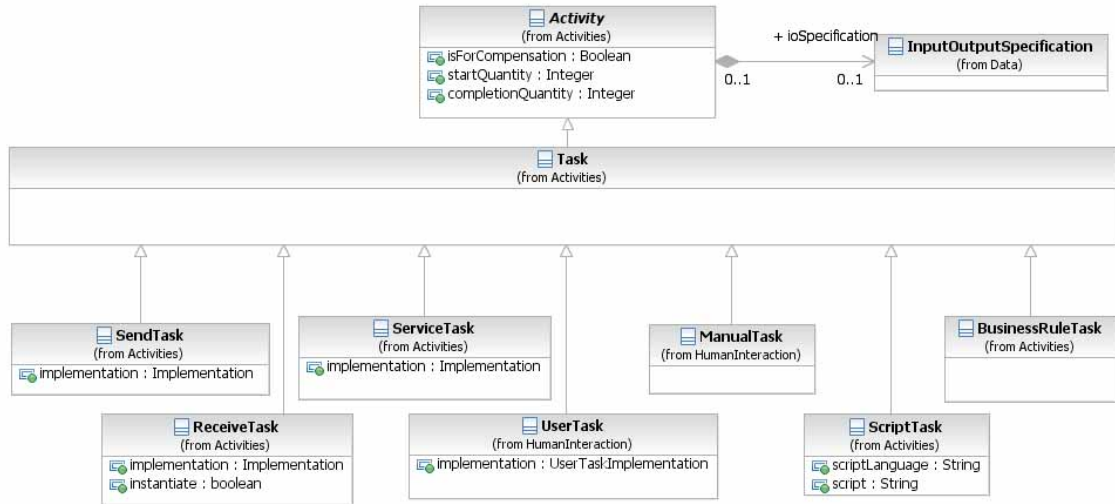


Figure B.8: BPMN 2.0 Task class diagram



Figure B.9: BPMN 2.0 ServiceTask class diagram

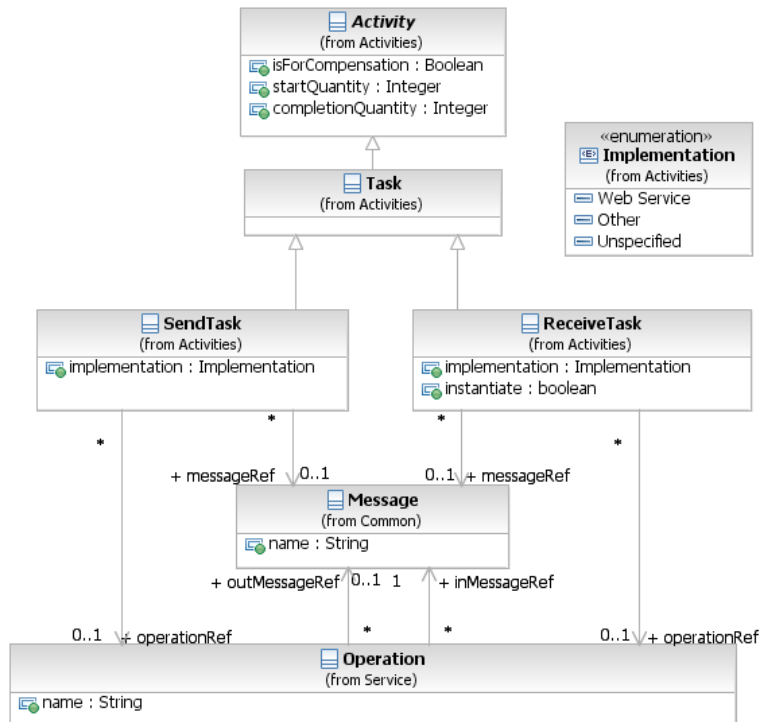


Figure B.10: BPMN 2.0 SendTask and ReceiveTask class diagram

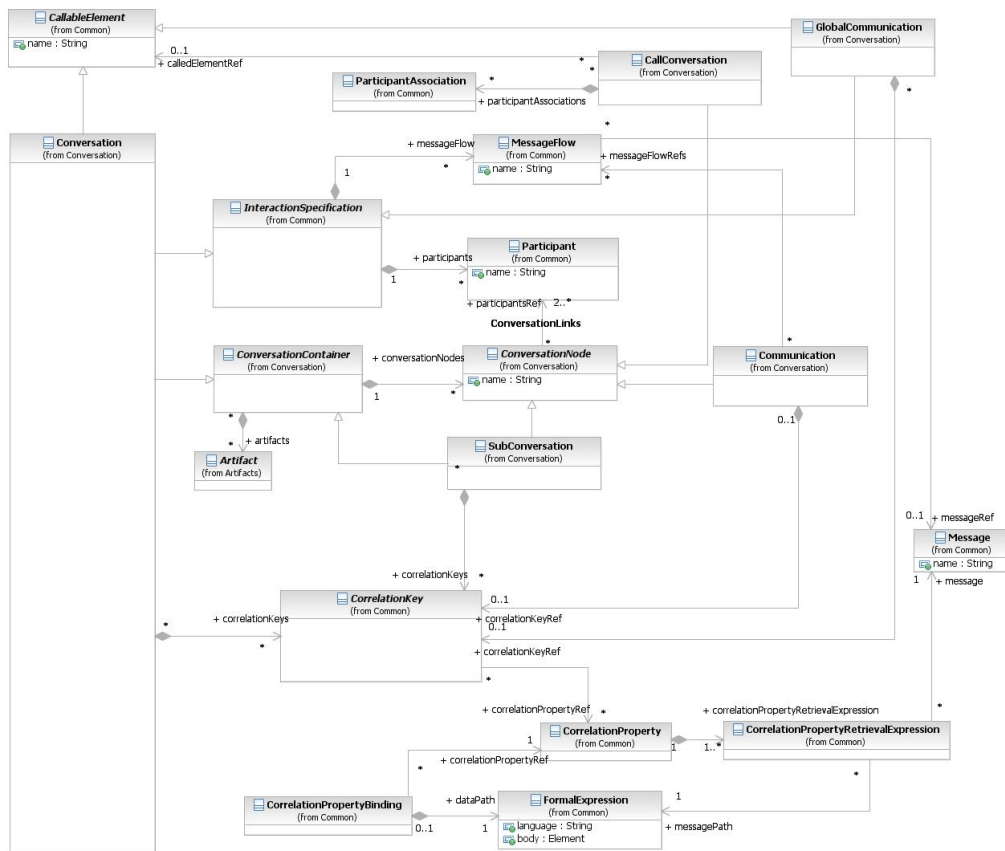


Figure B.11: BPMN 2.0 Conversation diagram class diagram

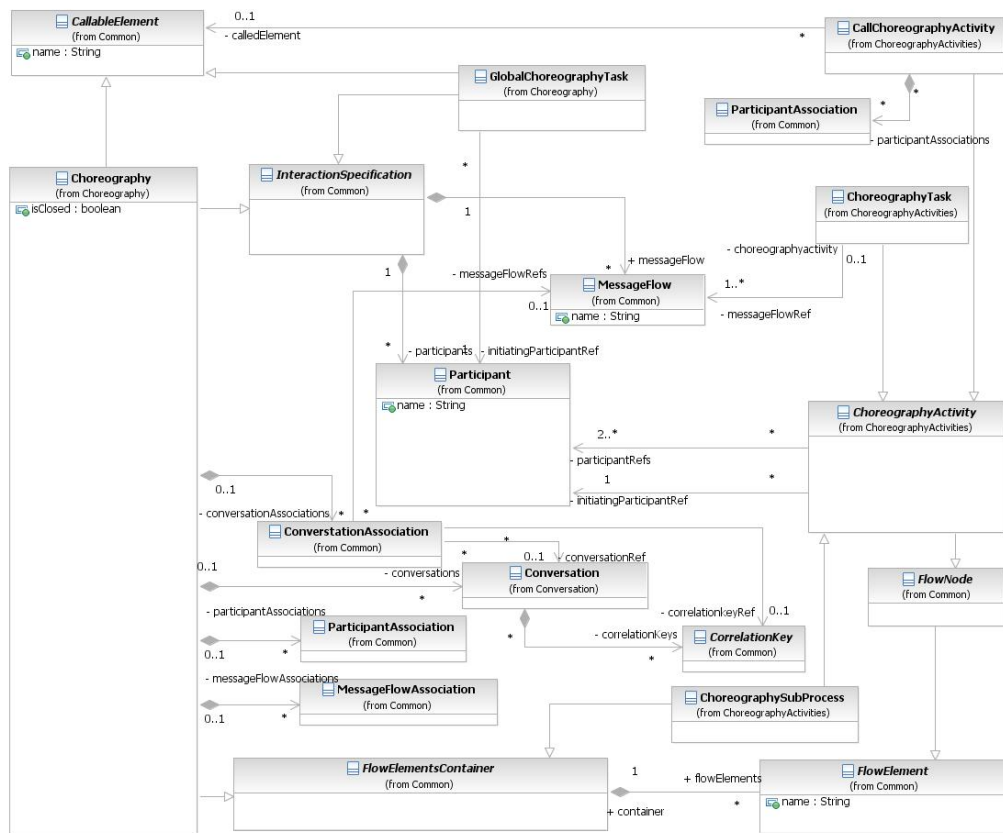


Figure B.12: BPMN 2.0 Choreography diagram class diagram

Athena CBP with BPMN 2.0

This chapter explores and discusses the possibility to use BPMN 2.0 as a modelling language in a research project named ATHENA, a Cross-organizational Business Process (CPB) framework. Especially with the new diagram types Choreography and Conversation.

C.1 Background

“ATHENA is an Integrated Project partially funded by EU. The objective is to develop solutions to overcome today’s interoperability problems.”¹ Back in 2004, BPMN 2.0 as a specification did not exist. The background for this article is to explore and discuss which parts of ATHENA that BPMN 2.0 could replace.

C.2 Problem and research questions

It is an increasing trend inside organizations themselves, and between organizations, that systems and personnel cooperate across business boundaries and between divisions or sections inside the business. A business organization modelling framework should therefore incorporate the ability to model local business processes and at the same time be able to model cross organizational dialogue. One might want this framework to enforce visibility modifiers, like public and private, so that parts of a process can easily be shielded from others not concerned.

¹<http://www.sintef.no/Home/Information-and-Communication-Technology-ICT/Cooperative-and-Trusted-Systems--/Projects/ATHENA/> Online; 15-07-2009

Can BPMN 2.0 replace some of ATHENA CBP framework, and in what degree?

C.3 Method

The goal in this paper is to look at the ATHENA framework and BPMN 2.0 to find similarities and differences between them. The base of the discussion is the specification papers for BPMN 2.0 and ATHENA project report discussed above. By reading and comparing specifications side by side I intend to find and focus on the common functionalities in both ATHENA and BPMN 2.0.

C.4 Scenario

An often used scenario in cross-organizational business modelling is the buyer/seller scenario. The example is suitable for many reasons: Everyone is somewhat familiar with it, and it easily illustrates the need for graphical illustration of the communications process between the participants. The number of participants can easily scale up to make the details in the interactions go from very simple to very complex.

In a buyer/seller chain there can be several participants. Limitations are set by the modeller or the process itself. Defining a scope is important so that the part of the process that is to be modelled is modelled at the right detail. For instance, if you model the communications between a convenient store and its suppliers of milk, you probably don't need to incorporate in the model who is putting the milk in the refrigerator. But who or how you keep track of expiration dates on the milk might be more important. These aspects are all up to the modeller and the participants, and should be discussed in advance.

C.5 ATHENA CPB

The Cross-Organisational Business Process (CBP) framework focuses on modelling business processes and the interactions between different business processes. The framework was developed in the ATHENA research project (3). The framework defines a two-dimensional view; the vertical is the abstraction layer (business-, technical and execution layer or CIM, PIM and PSM levels) and the horizontal is the degree of visibility of the process (private process, view process and CPB).

C.5.1 The dimensions

The vertical dimension in CPB specifies the stakeholders of the abstraction layer in question.

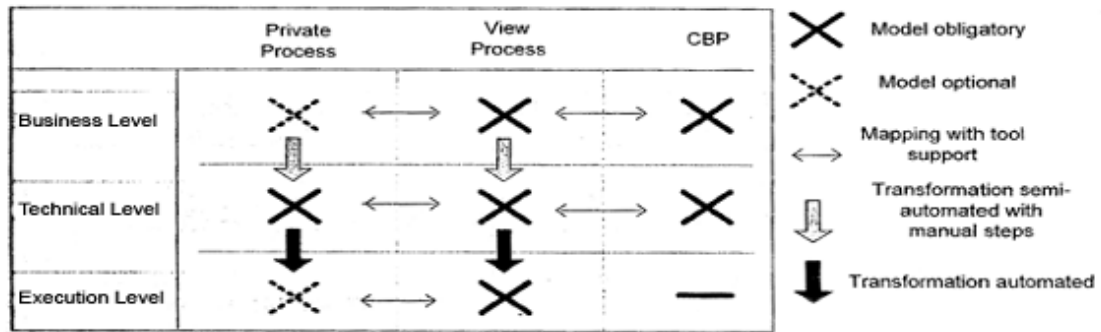


Fig. 2. Modelling Framework

Figure C.1: Modelling Framework

(13)

Business level: This level represents the business view on the cooperation and describes the interaction of the partners. The CBPs modelled on this level allows for analysing business aspects, like costs, involved resources etc (13).

Technical level: This level provides a more detailed view on the CBP representing the complete control flow of the process in a platform independent manner. Non-executable tasks are not considered. Also the message exchange between single tasks is modelled and can be analyzed. This supports reuse of the process models as they can be ported to various execution platforms (13).

Execution level: On this level the CBP is modelled on the modelling language of a concrete business process engine. It is extended with platform specific interaction information (13).

The Business level is in this context the same as a Computation Independent Model level (CIM). As with the Technical level is Platform Independent Model (PIM) and Execution level is a Platform Specific Model (PSM).

The horizontal axis of the two dimensions in CBP are designed to allow hiding of parts of processes that are considered private to the business.

CBP column is the model of the interactions between two or more participants.

Private process is the process of a specific participant. This process has the highest level of detail.

View process are the outline for several private processes typically showing the connection points for interaction and the sequence of these.

By giving different levels of visibility to the processes, businesses can easily hide private parts of a view process for the CBP. An example of a private process is discount calculation, or pricing. A business might not want discount rates to be public information but at the same time wants the model to show when price calculation occurs. So the view process will only show “price calculation”, but the private process might show the whole calculation.

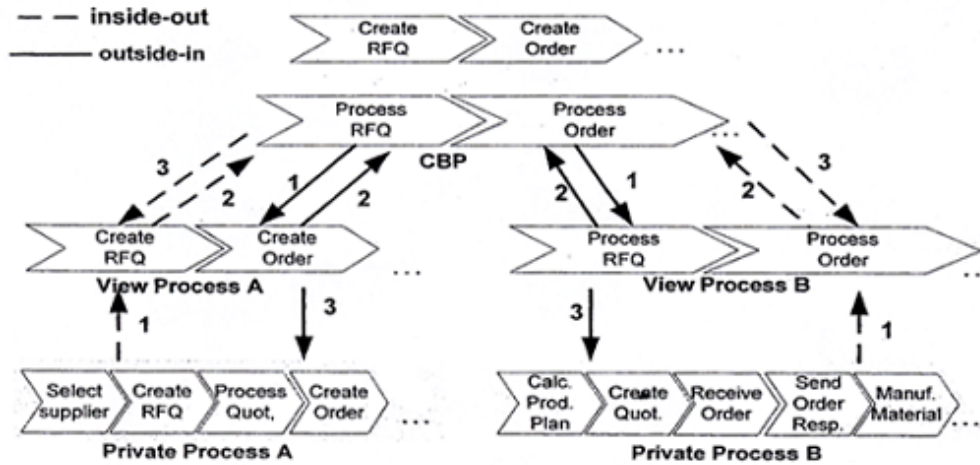


Fig. 3. Modelling procedures (outside-in vs. inside-out) illustrated with processes from the eProcurement scenario

Figure C.2: Modelling Procedures

(13)

C.5.2 Work methods

To model a CBP you need guidelines and procedures for the Collaboration between the participants. CBP incorporates two modes of doing this: Inside-out and outside-in. To choose which mode to use, you need to establish on what level you already are with regards to the processes. If you are about to develop a new interaction between participants, you might use the outside-in mode, and if you already have routines but need to model the existing routine, you use the inside-out mode.

Outside-in: You start by defining the CBP suitable for all partners at the highest level, e.g. the business level. You use an iteration method to create view processes for each participant and constantly change the CBP and the individual view processes (VP) until you have a CPB and a set of VPs you all agree on. All participants then create the private processes (PP) needed to fulfil the view processes according to the CBP.

Inside-out: Basically the opposite direction than the outside-in. Here you want to create a CBP on an existing process. Each participant creates PP and maps them to a view process. The sets of view processes are then combined to form a CBP using an iterative process until all participants can agree on the CBP.

“Both languages offer graphical notations which are easy to understand for business analysts and allow to analyze business aspects [...]” (13) MO2GO is a modelling tool that uses Integrated Enterprise Modelling (IEM) (29) as the modelling language, and ARIS uses Event-driven process chain (EPC). On the technical level CBP uses Maestro, an SAP tool giving more detail to the CBP. And finally, CBP uses the de facto standard BPEL (23) on the execution level.

CPB developed in the ATHENA research project is a complete framework from the business level (CIM) to the technical (PIM) level down to the executable (PSM) level.

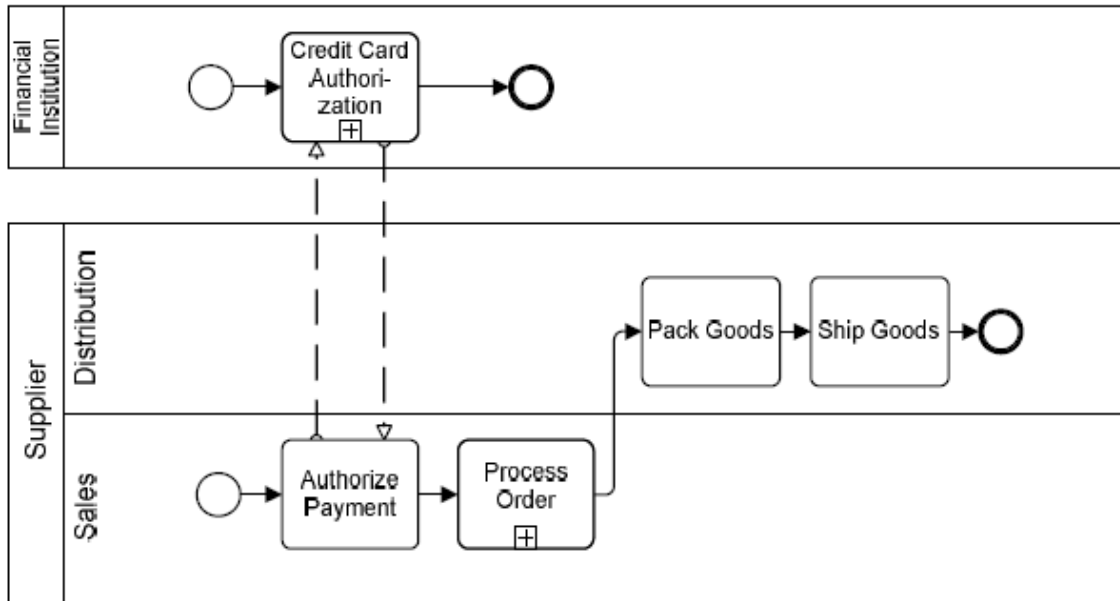


Figure C.3: BPMN 2.0 Collaboration example

(21)

Using well known modelling and execution platforms, CBP gives the business planners and the technical staff the right tools to develop and model cross-organizational processes.

C.5.3 MO2GO and IEM in CBP

IEM is object oriented and uses three main subclasses of a super class: Product, Order and Resource. Together they form an action class, which is what actually happens in the model. For example, if you need to model a buyer/seller example in IEM, you put the actual product requested as a product, the query for a product as an order and people handling the order for the product as resources. Those three combined forms an action in the process.

C.6 BPMN 2.0

Business Process Modelling Notation 2.0 (BPMN 2.0) is still an “initial specification for review and comment by OMG members.” (21, p. 26) In the current BPMN 1.x version there is a lack of cross business collaboration abilities that BPMN 2.0 is meant to address. By adding new diagrams called Choreography diagram and Conversation diagram, BPMN 2.0 will gain an extra level of abstraction, hopefully giving the modelling language the ability to model cross organizational relations. The new version also gives standardised persisting and interchange abilities to the language through support for XMI with the MOF based meta model (21).

C.6.1 Collaborations

BPMN 2.0 represent a participant using what is called a pool. Inside the pool, you can have several swim lanes, representing for instance different apartments inside the business or different processes. BPMN 2.0 gives no constraints on what level of abstraction you are suppose to use the different elements; it only specifies what is allowed to do with the classes specified in BPMN 2.0 meta model. So a participant can be everything between, say, corporations down to a specific person in a given apartment. Choosing the level of abstraction is important in BPMN 2.0, because you have full flexibility within the meta model as long you follow the constraints. Since you cannot build a hierarchy of pools, you can't model a business structure using pools (21).

C.6.2 Conversations

Conversation is the same as Collaboration, except that in a conversation it is not allowed to have processes in the pools, and all message exchanging is collapsed into single conversations. The idea behind this diagram is to give a bird's view of the whole chain, without the high detail information stored in a process. When modelling on Collaborations you find quite fast that a model will be big, so you have to divide different Collaborations in different diagrams. The Conversation diagram lets you put all Collaborations in one single diagram to see participation involvement with ease (21).

C.6.3 Choreographies

Choreography is a variant of collaboration. But, instead of focusing on how the participants perform their processing, a choreography “formalizes the way business Participants coordinate their interactions” (21). Choreographies can be interpreted as a business contract between the involved participants, displaying the direction and exact procedure the process is following. The specification presents a new diagram type that draws two or more participants as opposites on a rounded square, called a choreography activity. The name of each participant is written in the band opposite to each other, one of which is not shaded showing the initiator of the activity.

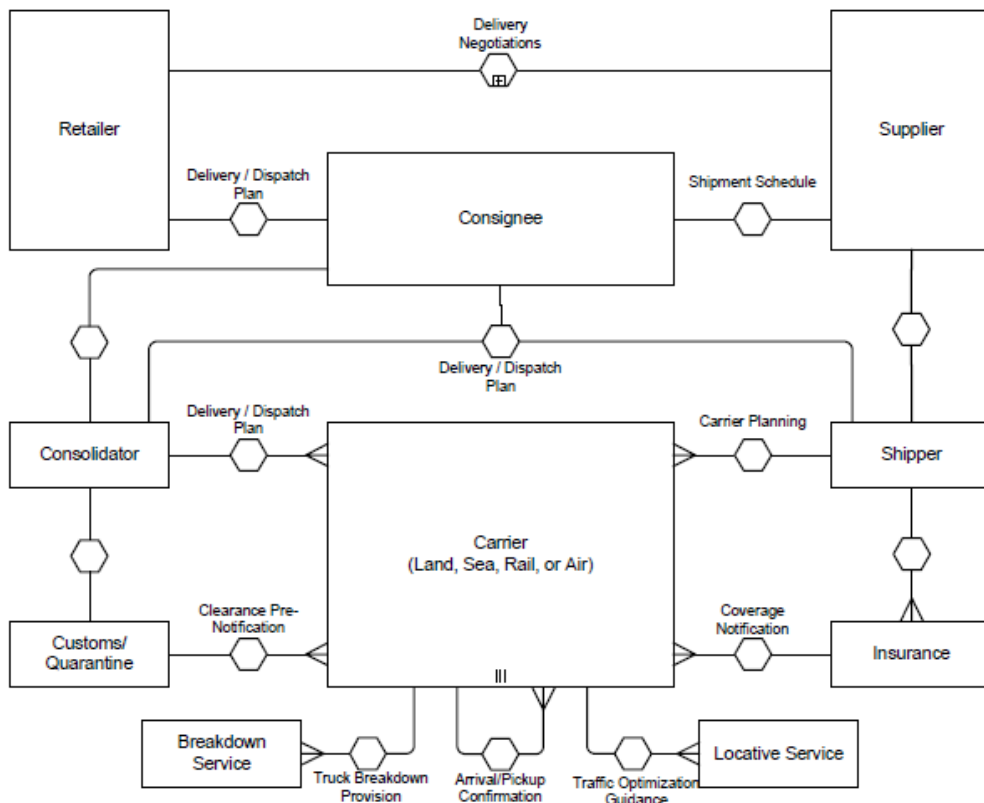


Figure C.4: BPMN 2.0 Conversation example

(21)

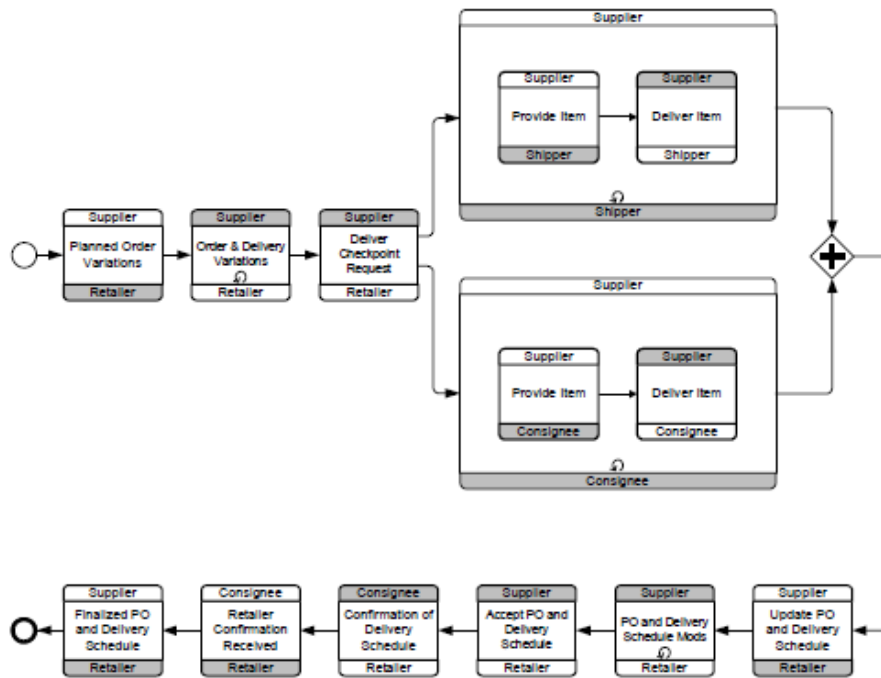


Figure C.5: BPMN 2.0 Choreography example

(21)

Activities are linked together using sequence flow arrows displaying the direction the process is suppose to follow (21).

C.7 Discussion

In CBP on the CIM-level called the business level and BPMN 2.0 is a CIM level process modelling language. The Conversation diagram of BPMN 2.0 could be used to model the CBP visibility column at the business level as we saw in figure C.1 on page 113. The Choreography diagram could be used for the view process of the framework, and the collaboration can then further detail the view process for private processes. This also clearly separates the model notation for each visibility column removing the need for specific modelling techniques for a single diagram notation.

ATHENA CBP use PIM4SOA to transform EPC- or IEM-model data created in ARIS or MO2GO from CIM to PIM level(13). For BPMN 2.0 Conversation diagram on the CBP visibility there are no transformation available down to PIM level. But both Choreography and Collaboration models for view and private process are created to be capable of execution by mapping to execution languages like WS-BPEL (20). This means that there might not be a strong need for transformation via PIM level. The PIM level could then be used only for architecture modelling, e.g. for web services, used in the BPMN 2.0 process and then executed in some system.

C.8 Conclusion

Many of the issues that ATHENA CBP addresses seems to have been a main focus for the specification of BPMN 2.0 and there might be indications that the BPMN 2.0 language together with a PIM language for information data modelling might cover many of ATHENA CBPs focus areas and can possibly replace the framework altogether.

Terms, definitions and abbreviations

This appendix section lists the some terms and abbreviations that are used in this thesis. Use this chapter as a reference while reading the document.

D.1 eXtensible Markup Language (XML)

XML is a type of markup language that supports semi structured data with an optional metamodel attached to validate the data. There a two types of XML documents: The data centric and the document centric. The first is used for machine readable documents, and the latter is used for human readable documents. The document centric type of XML is typically used for document standards. XML plays a big part in the W3C formats for present and future online communication standards, like OWL and RDF in representing semantics on the web and Soap used as a standard for WebService communication. And many others.

D.2 XML Metadata Interchange (XMI)

XMI is an interchange format used for sharing objects using XML. Sharing objects using XML is a comprehensive solution that builds on sharing data with XML.

D.3 Service Oriented Architecture (SOA)

Service Oriented Architecture is a way to describe and understand an organisation or a system to to maximize the agility, scale and interoperability. A service is

consumed and returns some type of value fulfilling the contract specified upfront. We only need to know what the service does, not how it does it. This enables us to perform tasks outside our own domain without knowing how it is done, but rather just how to initiate the work. SOA is then an architectural paradigm for defining how people, organisations and systems provide and use services to achieve results.

D.4 Object Oriented Role Analysis Method (OORAM)

OORAM was developed by Trygve Reenskaug and the method focuses on the analysis of processes to identify and describe interaction patterns between participants described as roles. By using this method and model the roles in the interaction, you abstract away any instances of objects and model at a higher level. Being a precursor to UML, OORAM uses many similar diagrams as UML does today and the role modelling in UML Collaboration diagrams stems from OORAM.

D.5 Object Management Group (OMG)

”OMG has been an international, open membership, not-for-profit computer industry consortium since 1989”¹. OMG maintains many open source specifications (UML, BPMN and many more).

D.6 Model Driven Architecture (MDA)

Model Driven Architecture is an initiative by OMG to provide language standards for use in model driven engineering. A language compliant with OMG’s MDA builds on a meta model defined with MOF.

D.7 Meta-Object Facility (MOF)

The Meta-Object Facility is an OMG standard for model-driven engineering and provides a meta data framework and services to enable the development and interoperability of such model and meta data driven systems. It is the model of the modelling language syntax. (22, p. 9)

D.8 Computation Independent Model (CIM)

A computation independent model, or sometimes called a business model, is the top most abstraction level in the three tier model life cycle. On this level, you model only business interactions and tasks.

¹<http://www.omg.org/gettingstarted/gettingstartedindex.htm>

D.9 Platform Independent Model (PIM)

A platform independent model is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms. (22, p. 9) Platform in this context means different hardware, different programming languages, deployment systems and such.

D.10 Platform Specific Model (PSM)

A PSM is a view of the system just as PIM, but with platform specific information on how a platform is to use the system. This can be the actual implementation in code, or modelled at an execution level, like BPEL.

D.11 Unified Modeling Language (UML)

UML is a visual language for specifying, constructing and documenting a system. It is a general purpose modelling language and can be used with all major object and component methods. (19)

