

Service and Software Architectures, Infrastructures and Engineering

Small or Medium-scale Focused Research Project

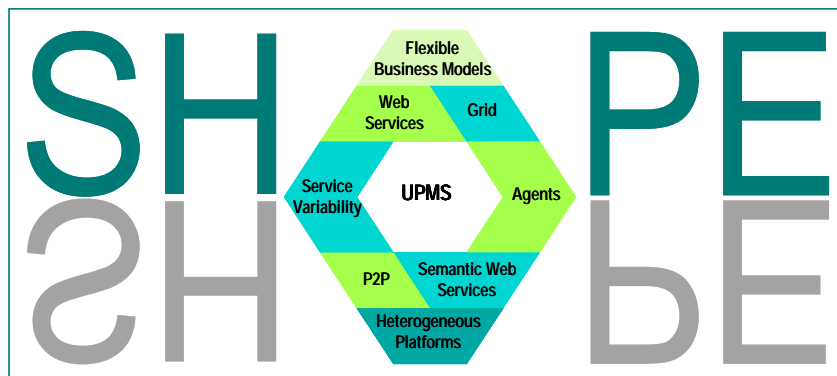
Semantically-enabled Heterogeneous Service Architecture and
Platforms Engineering

Acronym

SHAPE

Project No

216408



Deliverable D5.4

Model transformations from business models to
UPMSHA – Final Version

Work Package 5

Leading partner: DFKI

Editor: Christian Hahn

Authors: Christian Hahn, Dima Panfilenko

Dissemination level: Public

Date: 30 May 2010

Version: 1.0 - final

Versioning and contribution history

Version	Description	Contributors
0.1	Baseline for further work	Christian Hahn, DFKI
0.2	Input on model transformation from CIM to PIM level	Dima Panfilenko, DFKI
0.3.	Added conclusion, summary, and introduction	Christian Hahn, DFKI
0.4	Internal review	Brian Elvesaeter, SINTEF
0.5	Internal review	Gorka Benguria, ESI
0.6	Reviewers comments answered	Dima Panfilenko, DFKI
0.7	Review comments addressed	Christian Hahn, DFKI
1.0	Final editing	Christian Hahn, DFKI

Executive Summary

This report presents the Deliverable 5.4 "Model transformations from business models to UPMSHA – Final version" of the SHAPE project, including the final results of task T5.2 as defined in the Description of Work [3].

This deliverable presents the final status of the model transformations from CIM to PIM of the SHAPE methodology. It builds upon the results of the preceding Deliverable D5.1 [1] and D5.2 [6], which presented (i) the overall model transformation and deployment architecture that is refined in this deliverable to reflect the final status of the SHAPE project and (ii) gives an initial overview on the CIM to PIM transformations of the model transformation architecture.

The transformations developed between the CIM and PIM level are the following:

- CIMFlex to ShaML
- BPMN to ShaML
- CIMFlex to UML

Accompanying this report is the prototype implementation of the model transformations. The implementation is available at <http://www.shape-project.eu/download>.

The following main progress and results have been achieved since the interim version of the CIM to PIM model transformations as presented in [6]:

- Refinement of the model transformation between BPMN and SoaML
- Final integration into the SHAPE tool suite
- Publication of the research results in a scientific article (i.e. [7])

The allocation of this deliverable in the overall context of the SHAPE project is as follows: WP5 focuses on model transformation of the SHAPE methodology. This deliverable builds upon the results of the preceding deliverables D5.1 and D5.2, which outlined the overall approach for the SHAPE model transformations, and D5.2 that presented the initial version of the SHAPE CIM to PIM model transformations. The SHAPE model transformation architecture and its model transformations are based on the results of the technical development of metamodels in WP3. The transformations are part of the SHAPE methodology developed in WP2 and are integrated in the SHAPE Tool Suite (WP4). Moreover, they are applied in the industrial use cases of WP1.

Table of Contents

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
TABLE OF FIGURES	5
LIST OF TABLES	6
1 INTRODUCTION	7
1.1 OBJECTIVE OF THIS REPORT	7
1.2 STRUCTURE OF THIS REPORT	7
2 SHAPE MODEL TRANSFORMATION ARCHITECTURE	8
2.1 CIM TO PIM TRANSFORMATION.....	9
2.2 PIM TO PIM TRANSFORMATION.....	9
2.3 PIM TO PSM TRANSFORMATION.....	10
2.4 PSM TO CODE TRANSFORMATION	10
3 SAARSTAHL USE CASE ON CIM LEVEL	11
4 CIM TO PIM MODEL TRANSFORMATIONS	12
4.1 MODEL TRANSFORMATION: FROM CIMFLEX TO SOAML.....	12
4.1.1 CIMFlex BPMN to ShaML Mapping Set	12
4.2 MODEL TRANSFORMATION: FROM BPMN TO UML.....	17
4.2.1 CIMFlex BPMN to UML Activity Diagrams Mapping Set	17
4.2.2 CIMFlex Data to UML Class Diagram Mapping Set	21
5 ILLUSTRATIVE EXAMPLE: FROM CIM TO PIM	23
6 CONCLUSIONS AND OUTLOOK	24
7 ANNEX A: SERVICE-ORIENTED ARCHITECTURE MODELING LANGUAGE	25
ACRONYMS	26
REFERENCES	27

Table of Figures

Figure 1: Model transformation architecture.....	9
Figure 2: CIM-level CIMFlex BPMN diagram	11
Figure 3: Generated Saarstahl ParticipantArchitecture.....	23
Figure 4: The SoaML profile	25

List of Tables

Table 1: Task to ServiceInterface.....	Fehler! Textmarke nicht definiert.
Table 2: Sub-Process to ServicesArchitecture.....	Fehler! Textmarke nicht definiert.
Table 3: Pool to Participant / ServicesArchitecture.....	Fehler! Textmarke nicht definiert.
Table 4: Lane to Participant / Service Architecture.....	Fehler! Textmarke nicht definiert.
Table 5: Annotation (Message) to MessageType.....	Fehler! Textmarke nicht definiert.
Table 6: Message "Begin" to Service.....	Fehler! Textmarke nicht definiert.
Table 7: Message "End" to Request.....	Fehler! Textmarke nicht definiert.
Table 8: Role to Participant.....	Fehler! Textmarke nicht definiert.
Table 9: Roles to ServiceContract.....	Fehler! Textmarke nicht definiert.
Table 10: Lanes to ServiceContract.....	Fehler! Textmarke nicht definiert.
Table 12: Pool to Activity Partition.....	17
Table 13: Lane to Activity Partition.....	18
Table 14: Task to Action.....	18
Table 15: Start Event to Initial Activity.....	18
Table 16: End Event to Final Activity.....	18
Table 17: End Event to Flow Activity.....	19
Table 18: Parallel Decision to Fork Node.....	19
Table 19 Parallel Decision to Join Node.....	19
Table 20: Message Event to Signal Node.....	19
Table 21: Message Intermediate Event to Signal Node.....	20
Table 22: Event to AcceptEventAction.....	20
Table 23: Gateway to Decision Node.....	20
Table 24: Gateway to Merge Node.....	21
Table 25: Sequence Flow to Control Flow.....	21
Table 26: Association to Object Flow.....	21
Table 27: Message Flow to Object Flow.....	21
Table 28: Data Object to Class Object.....	22
Table 29: Attribute to Attribute.....	22
Table 30: Generalization to Generalization.....	22
Table 31: Association to Association.....	22
Table 32: Aggregation to Aggregation.....	22

1 Introduction

One of the core objectives of the SHAPE project is to promote **a new development paradigm with a higher degree of involvement of joint user and development communities** through minimising the gap between business and system modelling. In particular, this is done by a model transformation between the CIM and the PIM level. The creation of PIM model can be semi-automated from business models through model transformations defined in accordance to principles of Model-Driven Architecture. The PIM model is typically further refined to platform-specific models (PSMs) through automated model transformations

In this report, we give an initial overview of the model transformations between CIMFlex on the CIM level and ShaML on the PIM level of the SHAPE model-driven methodology. This is done by discussing the core model mappings in detail. Moreover, a detailed explanation on the updated model transformation architecture is given.

1.1 Objective of this report

The objective of this deliverable is to provide the details on the SHAPE relevant model transformations. In particular, this deliverables deals with the model transformations between CIMFlex on the CIM level of the SHAPE methodology and ShaML.

Consequently, the following model transformations are discussed in this deliverable:

- CIMFlex to ShaML
- BPMN to ShaML
- CIMFlex to UML

These model transformations are formalized within the SHAPE methodology and partially applied in the use case of the Saarlouis AG. This use case is used in this deliverable to illustrate the details of the model transformations.

1.2 Structure of this report

This report is structured as follows:

- **Chapter 2** reports on the slightly modified model transformation architecture.
- **Chapter 3** illustrates the use case models on the CIM level that define the source of the model transformations.
- **Chapter 4** specifies the CIM to PIM model transformations.
- **Chapter 5** illustrates the generated output of the model transformations based on the use case models illustrated in Chapter 3.
- **Chapter 6**, finally, concludes this deliverable.

In addition, the annexes contain supplementary material, namely:

- **Annex A** provides information on the details of the SoaML profile.

2 SHAPE Model Transformation Architecture

This section mainly deals with the model transformation architecture which is one of the main contributions of deliverable D5.1. The model transformation architecture will lay the foundation of the implemented model transformations as well as for the upcoming deliverables in work package 5.

The model transformation architecture is based on the ATHENA model-driven interoperability framework illustrated in [1]. It illustrates the core language used within the project, their relationship to the abstraction levels CIM, PIM and PSM as well as their relationship to other languages defined through model transformations, either model-to-model or model-to-text.

In order to reflect the current state of the SHAPE project, we slightly modified the model transformation architecture (see Figure 1). For this purpose, we removed peer-to-peer and grid platforms. Furthermore, we removed the Web Service Architecture on the PIM level. The transformations to web services are now directly addressing XSD, BPEL and WSDL.

On the highest level, business models encompass business rules, processes, services and other issues such as contracts involving humans and organizations to achieve business goals. These conform to the metamodel of CIMFlex. The middle layer contains the results of the proposal as transformation engines, extended SOA models, the standardized UPMS (SoaML) and extensions for semantically-enabled heterogeneous architectures (ShaML). This architecture allows the realization of one of the main goals of SHAPE namely to provide a transformation engine that maps business models to SOA/SHA models which are then transferred to the various execution platforms.

The transformation engine should also support visualization of services in business models (provide transformation support both ways). The PIMs will further be extended by concepts that allow smooth transformation to heterogeneous platforms and thus are extended SOA models (ShaML). The focus of SHAPE is on this layer. The lowest level shows the generated PSMs and their metamodels. The proposal will support model transformation for web services and semantic web services, as well as multi-agent systems. The architecture thus provides an integrated solution for service development that covers the life cycle of services from business goals and requirements to platform specific models for various platforms.

In the remainder of this section, we discuss the model transformation architecture in more detail by focusing on the abstraction levels, their core technologies as well as the model transformation needed. For this purpose, we distinguish between CIM to PIM transformations, PIM to PIM transformations, PIM to PSM transformations and finally, PSM to Code transformations. The complete SHAPE model transformation architecture is depicted by Figure 1.

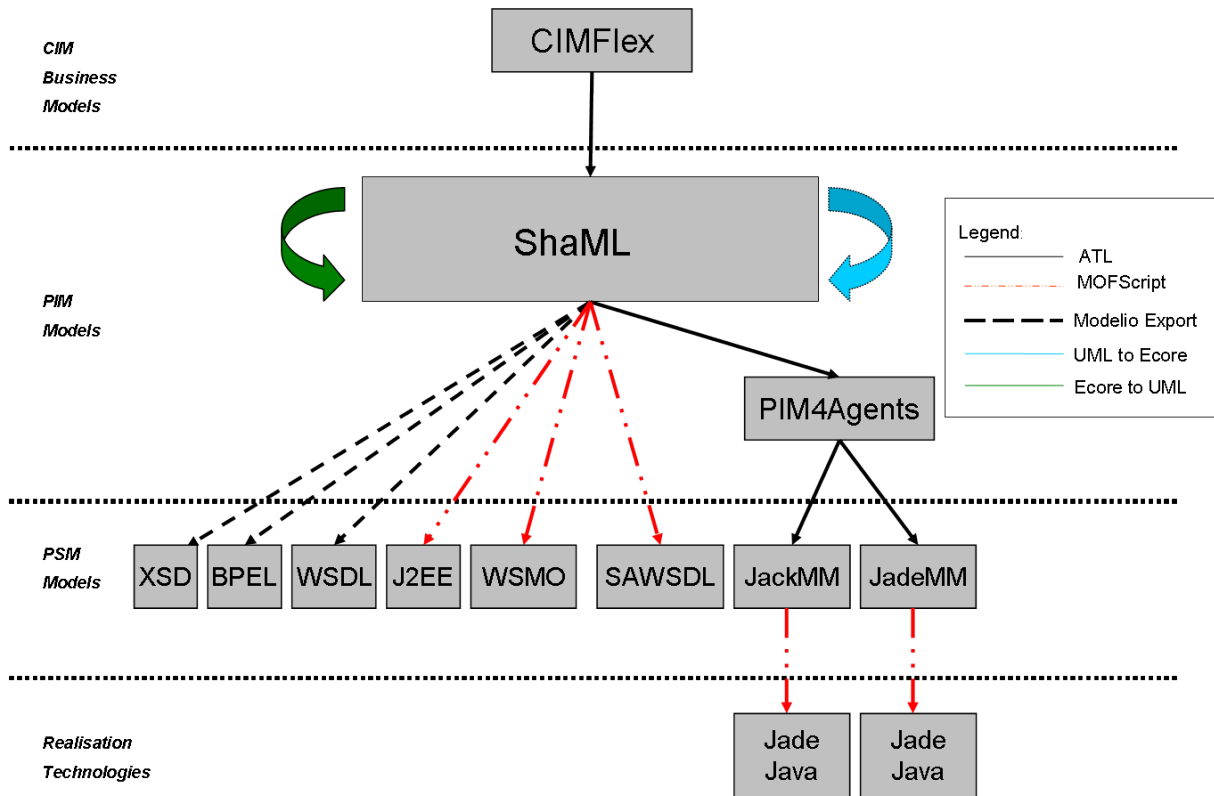


Figure 1: Model transformation architecture

2.1 CIM to PIM transformation

- Model transformation between CIMFlex and ShaML:** Computation independent models (CIM) focus the business context of a system and represent the business requirements in a semi-structured form by means of a notation with predefined symbols. The challenge in transforming CIMFlex models to SoaML is to generate the appropriate system relevant constructs for SoaML according to the generic business context on CIM level. CIMFlex supports in its initial version the model-to-model transformation by making use of the Atlas Transformation Language (ATL). A detailed description on the CIMFlex to ShaML transformation is given in this deliverable.

2.2 PIM to PIM transformation

- Model transformation between ShaML and PIM4Agents:** Transferring ShaML models into PIM4Agents models is done through a model-to-model transformation. For this purpose, the transformation engine Atlas Transformation Language (ATL) is used. The detailed mapping rules are discussed in [4].
- Transformation between ShaML (SoaML) Metamodel and ShaML (SoaML) Profile:** This transformation is necessary as the model transformation between CIMFlex and SoaML generates a model that is transferred to the SoaML Profile to use it as input for the ShaML to PIM4Agents transformation. Details on the mappings are given in [4],
- Transformation between ShaML (SoaML) Profile and ShaML (SoaML) Metamodel:** This transformation is necessary to include variability into the SoaML profile. Details on the mappings are given in [4].

2.3 PIM to PSM transformation

- **Model transformation between PIM4Agents and JackMM:** The model transformation between the PIM4Agents metamodel and the metamodel of Jack (JackMM) has been presented in [1].
- **Model transformation between PIM4Agents and JadeMM:** The model transformation between the PIM4Agents metamodel and the metamodel of JADE (JadeMM) has been presented in [1].
- **Model transformation between ShaML and WSMO:** The model transformation between the metamodel of SoaML and WSMO is specified through a model-to-text transformation using the MOFScript engine. Detailed mappings are discussed in [4].
- **Model transformation between ShaML and SAWSDL:** The model transformation between the metamodel of SoaML and WSMO is again specified through a model-to-text transformation using the MOFScript engine.
- **Model transformation between ShaML and WSDL, XSD and BPEL:** These transformations are provided by the modeling environment of Modelio. Details on the transformations are given [4].
- **Model transformation between ShaML and J2EE:** This model transformation is also specified using the MOFScript language.

2.4 PSM to Code transformation

- **Model transformation between JackMM and Code:** The generated Jack models conforming to the JACK metamodel (JackMM) are in a final step mapped into specific JACK code that can be imported by the JACK IDE. For this purpose we defined a model-to-text transformation using MOFScript which uses the generated JackMM models as input and produces the JACK-specific Gcode. When imported, the generated Gcode can easily be transferred into Java using the facilities provided by Jack and executed. In [1], a summary on PSM to Code transformations is given.
- **Model transformation between JadeMM and Code:** Like described in the case of JACK, for JADE we also defined a model-to-model transformation using MOFScript. However, in this case, we defined a transformation directly from JadeMM to Java code that can finally be executed. In [1], a summary on PSM to Code transformations is given.

In this report, we focus on the CIM to PIM transformations by discussing the model transformations from CIMFlex to ShaML and BPMN to ShaML.

3 Sairstahl Use Case on CIM level

The CIMFlex editor allows the user to create and refine a semi-formal model of a business process, an organisational structure, a data structure or business rules based on the input coming from the domain users. The editor is able to create, change and store these types of models in EPC or BPMN notation [10], [11]. As storage format XML files are generated. The target users of this component are the domain user and especially the business analysts. From an architectural point of view the component has two interdependencies with other components for its output. The information, which is required for the creation of a CIM model, will be derived from the use cases by the domain users. The output of the CIM level editor can have two different forms depending on its purpose. On the one hand, a model on CIM level in BPMN notation can be used as the technical information description draft, giving a starting point for the transformation into BPEL for further execution of the resulting model or the enrichment with further technical information. On the other hand the output of the CIM level editor is the starting point for the CIM to PIM transformation. In this case the editor does not provide the models in BPMN notation, but transforms them into ShaML models. The conceptual and technical details of this transformation are described in the Section 4. The prototypes of this transformation are stored on the SHAPE website [11].

Figure 2 depicts the Sairstahl AG architecture, In this example, there is a pool which represents the Sairstahl architecture and a set of lanes within it which represent the different entities involved in the overall supply chain. The entities are Sales Department, Order Department, Tech Inspection etc. Furthermore there are tasks within the lanes which stand for workflow activities and sequence flows between them which symbolize the work order. Every time the workflow changes the lane, it is considered that an interaction between the two Participants represented by lanes takes place

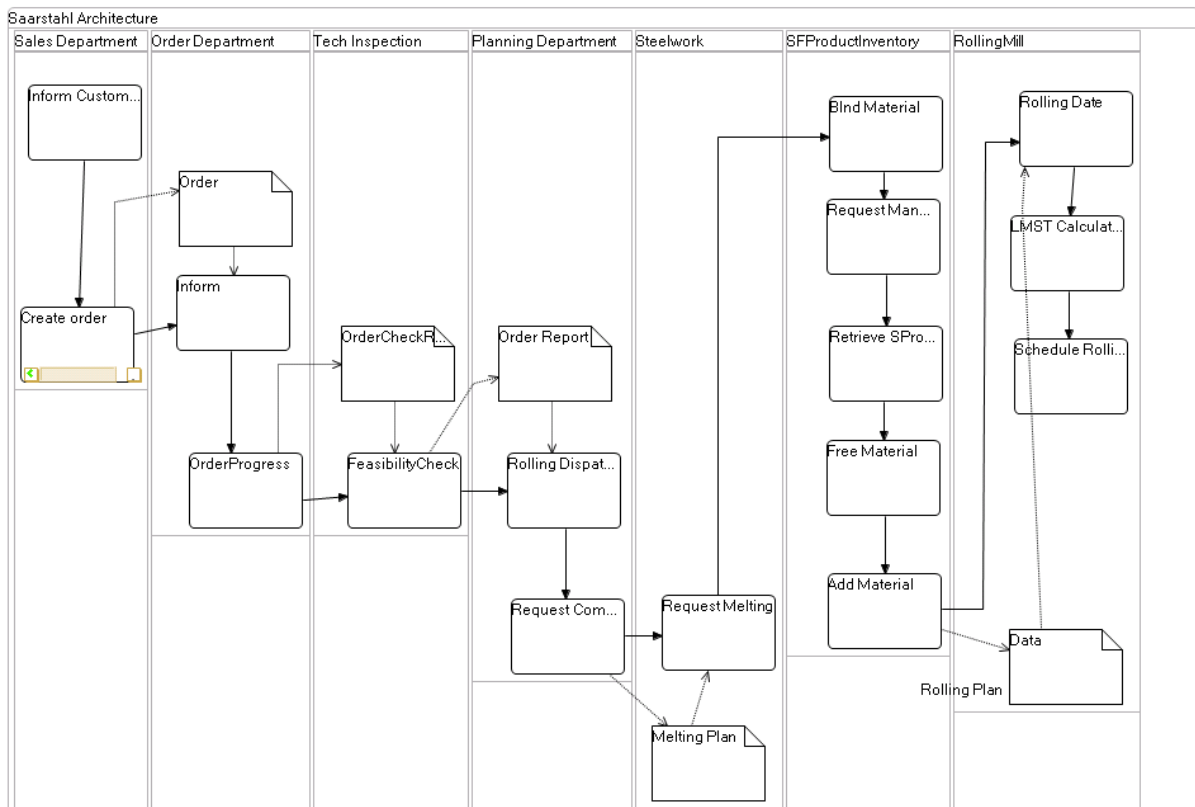


Figure 2: CIM-level CIMFlex BPMN diagram

4 CIM to PIM Model Transformations

4.1 Model Transformation: From CIMFlex to SoaML

In this section we provide some aspects of the high CIM-level service modelling with the aid of the BPMN. This notation is well-known and established since the beginning of the 21st century, moreover it has been standardized and there are more than 50 products, both commercial and open-source, providing the implementation of this standard. The particular considerations with respect to modelling services by the business users are that there is a little awareness of the services by CIM-level users, on the one hand, and even if there would be any knowledge about it, there are no direct constructs describing the services on the CIM-level in the BPMN notation anyway. Of course the upcoming BPMN 2.0 standard includes the services modelling and the according constructs for it, but it only rules out the second, more technical problem, and not the first one – understanding.



For the solution of this problem we propose a semi-automated approach in this section based on a model-to-model (M2M) transformation from CIM-level BPMN models to PIM-level SoaML-based models. Those models on the higher abstraction level in BPMN would be analysed through a set of mapping rules and would result in a service model representing according constructs and architectures needed for the comprehensive PIM-level model as a basis for the further transformation to the PSM-level. The further section content comprises the manufacturing example and the mapping rules identified and needed for the services mapping from CIM- to PIM-level models. In addition there are technical details of the transformation presented for the BPMN to ShaML mapping set giving a short insight into the serialisation of the models during transformation.

4.1.1 CIMFlex BPMN to ShaML Mapping Set

Mapping Rule 1: Task to ServiceInterface

As a task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process, it can be then mostly closely assigned to Action construct in this mapping, as it gives the abstract interface for the job done and at the same time does not give further specification of the workflow implementing this task. In CIM manufacturing example it means all three Tasks “Prepare Order”, “Purchase” and “Receive Order” are mapped to Actions. Table 1 illustrates the mapping of the notation.

Table 1: Task to UML Action

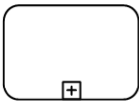

BPMN Description	Symbol	ShaML Description	Symbol
Task		UML Action	

Mapping Rule 2: Sub-Process to ServicesArchitecture

As a sub-process represents a more complex process than a simple task, but still can be seen as a whole, it can be assigned to the role if the counterpart of the ServicesArchitecture. It should be mentioned, though, that this ServicesArchitecture is not the bottom level and can be subdivided further (through roles). Table 2 illustrates the mapping of the notation.

Table 2: Sub-Process to ServicesArchitecture



BPMN Description	Symbol	ShaML Description	Symbol

Sub-process		Collaboration with the stereotype ServiceArchitecture	
-------------	---	---	---

Mapping Rule 3: Pool to ServicesArchitecture

A pool in BPMN stands for a business entity or a participant of a process, on the one hand. It also can be structured with respect to further Participants of the process, thus creating a participants' hierarchy. These two points together put the pool on a role of a candidate for a Participant or ServicesArchitecture, depending on the modeller's intention. Table 3 illustrates the mapping of the notation.


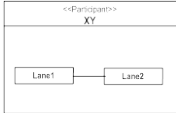
Table 3: Pool to ServicesArchitecture

BPMN Description	Symbol	ShaML Description	Symbol
Pool		Collaboration with the stereotype ServiceArchitecture	

Mapping Rule 4: Lane to Participant

A lane represents a participant or a department in BPMN and is situated in a pool, thus showing the two-tier hierarchy. In order to show the possibility for further subdivision (which is also ongoing in the current BPMN2 proposals), the lane is first mapped to a Participant and next tiers of this hierarchy are constructed using the role constructs described below. Table 4 illustrates the mapping of the notation.

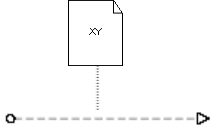

Table 4: Lane to ServicesArchitecture

BPMN Description	Symbol	ShaML Description	Symbol
Lane		Component with the stereotype Participant	

Mapping Rule 5: Annotation (Message) to MessageType

This Annotation construct in BPMN is connecting two participants, from which one is providing and another is consuming the service in question. As the service provided is not described extensively in BPMN, it is not feasible to get to a ServiceContract message content exactly, so the next abstraction would exactly refer to the MessageType, thus providing the mechanism for reflecting the "loose" messages between different participants. Table 5 illustrates the mapping of the notation.


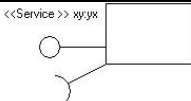
Table 5: Annotation (Message) to MessageType

BPMN Description	Symbol	ShaML Description	Symbol
Annotation (Message)		Class with the stereotype MessageType	

Mapping Rule 6: Message “Begin” to Service

The beginning point of each and every message in BPMN has the following semantics – it should be the starting end of the data channel between two participants or pools. This exact meaning also has the Service point in ShaML, which finds its accordance in this mapping point. The Participants in ShaML are using this construct in order to provide services for other participants in the modelled architecture. Table 6 illustrates the mapping of the notation.


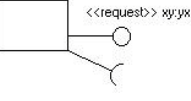
Table 6: Message “Begin” to Service

BPMN Description	Symbol	ShaML Description	Symbol
Message “Begin”		Service	

Mapping Rule 7: Message “End” to Request

The ending point of each and every message in BPMN has the semantics that looks very alike with the message beginning point, but is situated on the other end of the communication channel. The similar semantics of the Request point in ShaML offers this construct to be mapped to the messaging end from the BPMN. The aim of this mapping is the reflexion of the data channel target in the service consumption of the modelled architecture. Table 7 illustrates the mapping of the notation.



Table 7: Message “End” to Request

BPMN Description	Symbol	ShaML Description	Symbol
Message “End”		Request	

Mapping Rule 8: Role to Participant

As mentioned before, the role is used for structuring the process participant architecture, thus giving further possibilities for modelling the participants in the BPMN process view. This construct also provides the interface to the role view in CIM-level, where more details could be modelled, which are abstracted in the BPMN process view. Table 8 illustrates the mapping of the notation.

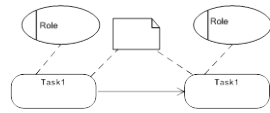
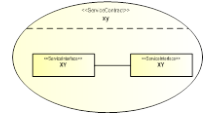
Table 8: Role to Participant

BPMN Description	Symbol	ShaML Description	Symbol
Role		Component with the stereotype Participant	

Mapping Rule 9: Roles to Service Contract

The ServiceContract construct is a complex one even in ShaML itself. There is also no single construct in CIM-level representing this entity, but rather a certain pattern of objects: this are two single tasks following one after another in a container, connected with a sequence flow and associated with one data object. See Mapping Rule 28 for data object mapping details. Table 9 illustrates the mapping of the notation.

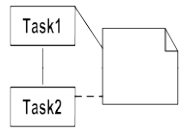
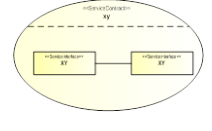
Table 9: Roles Pattern to ServiceContract

BPMN Description	Symbol	ShaML Description	Symbol
Role1 → Role2		Collaboration with the stereotype ServiceContract	

Mapping Rule 10: Lanes to ServiceContract

This transformation also reflects the service contract from the CIM level model into the ShaML and later into PIM for agent interaction specification. There is also a task sequence connected by a sequence flow, but the participants are represented through different lanes in the same pool, thus showing another possibility for a participant architecture modelling. The two tasks that belong to a ServiceContract also share a data object. Table 10 illustrates the mapping of the notation.




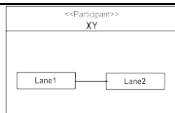
Table 10: Lanes Pattern to ServiceContract

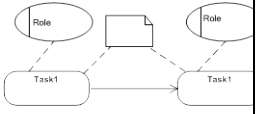
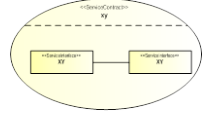
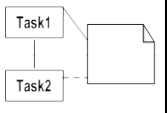
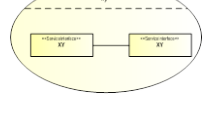
BPMN Description	Symbol	ShaML Description	Symbol
Lane1 → Lane2		Collaboration with the stereotype ServiceContract	

4.1.2 CIMFlex BPMN to ShaML Technical details of the ATL-implemented transformation

This section represents the short insight into technical realisation of this rule mapping set with aid of a simple example. The complete source code and comments could be found on the SHAPE website.

Table 11: Structure Transformation Patterns

BPMN Realisation	Symbol	UML/SoaML Realisation	Symbol
ProcessView		Model	
Pool		Collaboration mit Stereotype ServiceArchitecture	
Lane		Component mit Stereotype Participant	

Role1→ Role2		Collaboration mit Stereotype ServiceContract	
Lane1→ Lane2		Collaboration mit Stereotype ServiceContract	

1) General Characteristics of the Realisation

involved Rules:

- Metamodel() produces the model.
- Pool2ServiceArchitecture() transforms pools into services architectures.
- createServiceContracts() transforms the construct Role1->Role2 into service contracts.
- createServiceContracts1() transforms the construct Lane1->Lane2 into service contracts.
- lane2participant() transforms lanes to participants.
- association2association() transforms CIMFlex-associations to SoaML-associations

involved Helpers:

- checkAssociations() gives the amount of the CIMFlex-associations associated with Role1->Role2 back.
- checkAssociations1() gives the amount of the CIMFlex-associations associated with Lane1->Lane2 back.
- checkDataObjectsColUse() gives the amount of the CIMFlex-data objects associated with Role1->Role2 back.
- checkDataObjectsCol1() gives the sequence of the CIMFlex-data objects associated with Lane1->Lane2 back. It is possible, that the sequence has similar objects. The amount of objects is equal the amount of service contracts.

realisation:

1. In step one the transformation creates the model. It's the top element, which contains all other elements. In CIMflex BPMN the model equals the process view construct.
2. In step two the using of the SoaML-Profile is realised.
3. All elements of the model are designed as packaged elements. A model can contain associations, participants, service contracts and services architectures.
4. With the help of checkAssociations() and checkAssociations1() the amounts of the associations are divided into three subsets: The amounts of associations for the transformation Role1->Role2 and Lane1->Lane2 and all other associations. The objects which you get with the help of these helpers are only for construct detection and not visible in the output file.
5. With help of the helpers checkDataObjectsCol1() and checkDataObjectsColUse() the amounts of the data objects are divided in two subsets: the amount of data objects suitable for the transformation Role1->Role2 and the amount of data objects suitable for the transformation Lane1->Lane2. These amounts can overlap. The objects which you get with the help of the helper are transformed into service contracts.

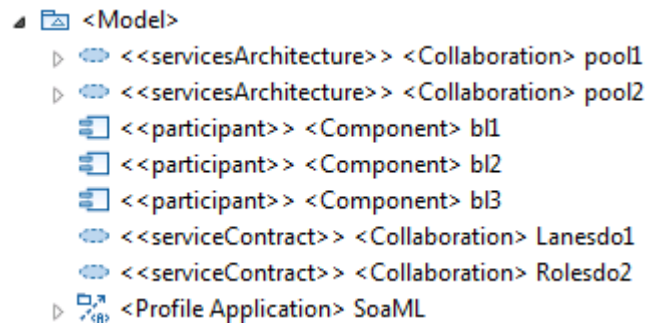


Figure 3 Object Hierarchy in SoaML Editor

structure of a *.soaml-Datei:

```

<soaml:Model>
<packagedElement ...></packagedElement>
<packagedElement ...></packagedElement>
<packagedElement ...></packagedElement>
<profileApplication ...>
  <appliedProfile href="..." />
</profileApplication>
</soaml:Model>
  
```

Thus we see how the transformation realisation is implemented at the level of the serialised model structure, which can be then used for more sophisticated mapping patterns implementation in the future work. More comments on that you can find on the SHAPE website in the WP5 prototype section.

4.2 Model Transformation: From BPMN to UML



The previous section dealt with the mappings between CIMFlex to ShaML. In the remainder of this section, a set of mapping rules will illustrate the transformation from BPMN to UML.

4.2.1 CIMFlex BPMN to UML Activity Diagrams Mapping Set

Mapping Rule 12: Pool2ActivityPartition

A BPMN-Pool is an Activity Partition in UML of the highest level. It is used as a container for other elements. A Pool/Activity Partition normally represents a workflow. Table 12 illustrates the mapping of the notation.


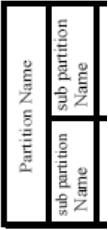
Table 12: Pool to Activity Partition

BPMN Description	Symbol	UML Description	Symbol
Pool		Activity Partition	

Mapping Rule 13: Lane to ActivityPartition

A Lane is an Activity Partition in UML as well, activity partitions can be nested, so you can create a similar pattern like the pool / lanes in the BPMN view. These constructs are responsibility areas of an actor executing the activities. Table 13 illustrates the mapping of the notation.


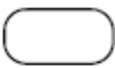
Table 13: Lane to Activity Partition

BPMN Description	Symbol	UML Description	Symbol
Lane		Activity Partition	

Mapping Rule 14: Task to Action

A BPMN Task represents an UML Action. A task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process. Table 14 illustrates the mapping of the notation.



Table 14: Task to Action

BPMN Description	Symbol	UML Description	Symbol
Task		Action	

Mapping Rule 15: StartEvent to InitialActivity

A BPMN Start Event represents an UML Initial Activity. The Start Event contains different event types, which give a detailed view of the event function. Table 15 illustrates the mapping of the notation.

Table 15: Start Event to Initial Activity



BPMN Description	Symbol	UML Description	Symbol
Start Event		Initial Activity	

Mapping Rule 16: EndEvent to FinalActivity

A BPMN End Event represents an UML Final Activity. The End Event contains different event types, which give a detailed view of the event function. There are no actions afterwards. Table 16 illustrates the mapping of the notation.

Table 16: End Event to Final Activity



BPMN Description	Symbol	UML Description	Symbol

End Event		Final Activity	
-----------	---	----------------	---

Mapping Rule 17: EndEvent to FlowActivity

This mapping rule applies any node indicating an end of a branch e.g. after merging, but not the end of the activity. Table 17 illustrates the mapping of the notation.


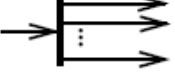
Table 17: End Event to Flow Activity

BPMN Description	Symbol	UML Description	Symbol
End Event		Flow Activity	

Mapping Rule 18: ParallelDecision to ForkNode

A BPMN Parallel Decision represents an UML Fork Node. The function of a parallel decision is the exact semantic accordance. It splits the control flow into multiple concurrent flows. Table 18 illustrates the mapping of the notation.



Table 18: Parallel Decision to Fork Node

BPMN Description	Symbol	UML Description	Symbol
Parallel Decision		Fork Node	

Mapping Rule 19: ParallelDecision to JoinNode

After splitting the control flow another parallel decision closes this splitting by joining the multiple concurrent flows together. After the parallel decision / join node the concurrent flows are synchronized. Table 19 illustrates the mapping of the notation.

Table 19 Parallel Decision to Join Node


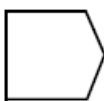
BPMN Description	Symbol	UML Description	Symbol
Parallel Decision		Join Node	

Mapping Rule 20: MessageEvent to SignalNode

This mapping rule is used for sending signals during the activity (event explained beyond). Table 20 illustrates the mapping of the notation.

Table 20: Message Event to Signal Node



BPMN Description	Symbol	UML Description	Symbol

Message Event		Signal Node	
---------------	---	-------------	---

Mapping Rule 21: MessageEvent to SignalNode

This mapping rule is used for sending signals at the end of the activity (end event explained beyond). Table 21 illustrates the mapping of the notation.

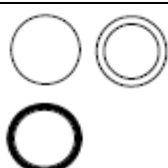

Table 21: Message Intermediate Event to Signal Node

BPMN Description	Symbol	UML Description	Symbol
Message Event		Signal Node	

Mapping Rule 22: Event to AcceptEventAction

Time Events or the Events that issue messages to other activities. Table 22 illustrates the mapping of the notation.


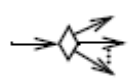
Table 22: Event to AcceptEventAction

BPMN Description	Symbol	UML Description	Symbol
Event		Accept Event Action	

Mapping Rule 23: Gateway to 2DecisionNode

A BPMN Gateway represents an UML Decision Node. The Gateway splits the work flow into several flows. Only the parallel gateway has another function. Table 23 illustrates the mapping of the notation.



Table 23: Gateway to Decision Node

BPMN Description	Symbol	UML Description	Symbol
Gateways		Decision Node	

Mapping Rule 24: Gateway to MergeNode

After splitting the workflow you need to join it again. Another Gateway / Merge Node is reserved for this event. Table 24 illustrates the mapping of the notation.


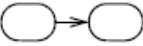
Table 24: Gateway to Merge Node

BPMN Description	Symbol	UML Description	Symbol
Gateways		Merge Node	

Mapping Rule 25: SequenceFlow to ControlFlow

A BPMN Sequence Flow represents an UML Control Flow. It is a connection between the two actions in one and the same BPMN-Pool. Table 25 illustrates the mapping of the notation.

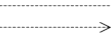

Table 25: Sequence Flow to Control Flow

BPMN Description	Symbol	UML Description	Symbol
Sequence Flow		Control Flow	

Mapping Rule 26: Association to ObjectFlow

A BPMN Association represents an UML Object Flow. It is a connection between a BPMN-Flow Object and a non-Flow Object. Table 26 illustrates the mapping of the notation.



Table 26: Association to Object Flow

BPMN Description	Symbol	UML Description	Symbol
Association		Object Flow	

Mapping Rule 27: MessageFlow to ObjectFlow

A BPMN Message Flow represents an UML Object Flow. It is a connection between a BPMN-Object in one Pool and an object in another Pool, possibly the Pool itself. Table 27 illustrates the mapping of the notation.

Table 27: Message Flow to Object Flow


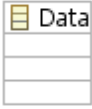
BPMN Description	Symbol	UML Description	Symbol
Message Flow		Object Flow	

4.2.2 CIMFlex Data to UML Class Diagram Mapping Set

Mapping Rule 28: DataObject to ClassObject

A CIMFlex Data Object represents an UML Class Object. It contains data values of the data type specified. This element is the source structure of the CIMFlex Data View. Table 18 illustrates the mapping of the notation.



Table 28: Data Object to Class Object

BPMN Description	Symbol	UML Description	Symbol
Data Object		Class Object	

Mapping Rule 29: Attribute to Attribute

A CIMFlex Attribute represents an UML Attribute. It is a specified data value, which is inside a Data Object. Table 29 illustrates the mapping of the notation.

Table 29: Attribute to Attribute

BPMN Description	Symbol	UML Description	Symbol
Attribute		Attribute	

Mapping Rule 30: Generalization to Generalization

A CIMFlex Generalization represents an UML Generalization. It is a connection between a specified and a generalized element. Table 30 illustrates the mapping of the notation.



Table 30: Generalization to Generalization

BPMN Description	Symbol	UML Description	Symbol
Generalization		Generalization	

Mapping Rule 31: Association to Association

A CIMFlex Association represents an UML Association. It is a connection between two elements. Table 31 illustrates the mapping of the notation.



Table 31: Association to Association

BPMN Description	Symbol	UML Description	Symbol
Association		Association	

Mapping Rule 32: Aggregation to Aggregation

A CIMFlex Aggregation represents an UML Aggregation. It is a weak connection between two elements. Table 32 illustrates the mapping of the notation.

Table 32: Aggregation to Aggregation

BPMN Description	Symbol	UML Description	Symbol
Aggregation		Aggregation	

5 Illustrative Example: From CIM to PIM

In order to represent the data these Participants are using for interaction there are DataObject elements connected to the source and target BPMN Tasks, thus giving the idea of the data flow between Participants and the notion of a ServiceContract on the PIM-level, for example PreProductionContract between SalesDepartment and Order Department. The Lanes representing Participants thus will be transformed into SoaML Participants and the interchanges between the lanes are here the ServiceContracts between the Participants.

Every component on the target PIM-level is modelled as a SoaML participant. Each participant offers several services. A service is instantiated by a class typed as Service Interface. The architecture itself encapsulates the internal services and offers an ordering service for the customers to the outside.

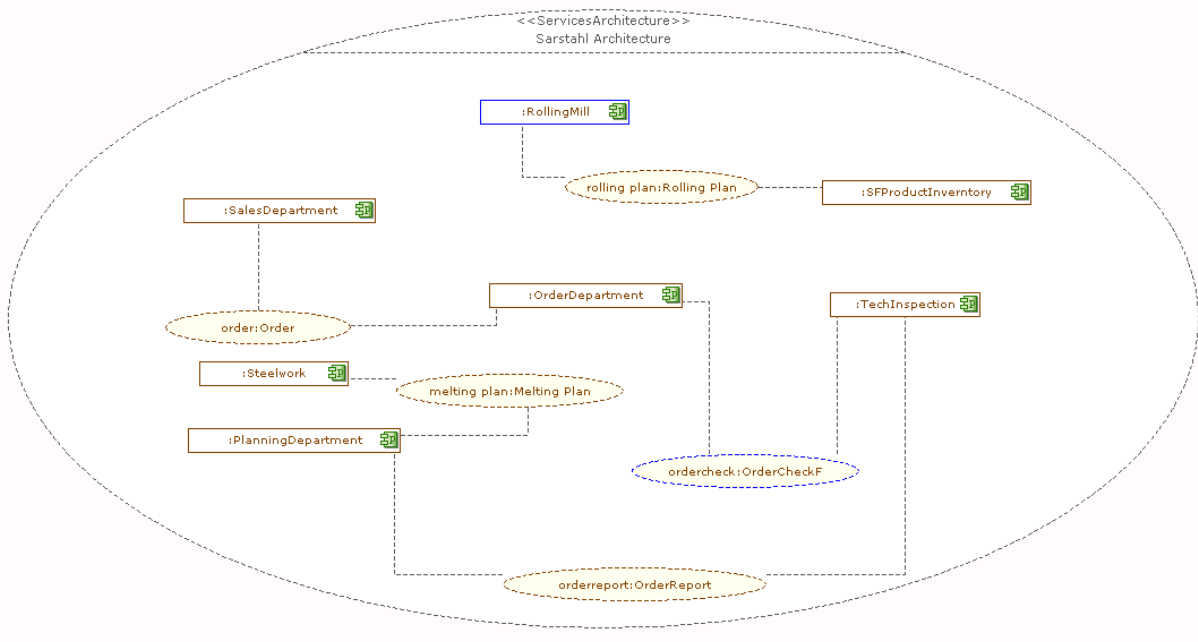


Figure 4: Generated Saarstahl ServicesArchitecture

6 Conclusions and Outlook

This deliverable presented the final version of the SHAPE model transformations from business models to SoaML. In particular, this deliverable focuses on the model transformation between CIMFlex representing the CIM level of the SHAPE model transformation architecture and ShaML on the PIM level. A detailed discussion on the model transformations between ShaML and the various execution languages on the PSM level of the SHAPE model-driven methodology is presented in [8].

For defining the model transformations, we presented the conceptual mappings between ShaML and the underlying platforms and illustrated the mappings using illustrative examples partly related to the use case of Saorstahl AG developed in WP1. The technical details were presented shortly, which give an insight into the principles of functioning of ATL and SoaML, for the prototypes of the transformations please refer to the SHAPE website [12].

The model transformation consists of two parts, i.e., CIMFlex BPMN to ShaML and CIMFlex BPMN to UML that were applied in the two use case scenarios of SHAPE. The CIM to PIM transformations were integrated into the SHAPE tool suite and published in a scientific article (i.e. [7]). The future work will embrace the research on the transformations in domains other than steel and oil industry sectors.

7 Annex A: Service-Oriented Architecture Modeling Language

The Service-Oriented Architecture Modeling Language (SoaML) is standardized in OMG. It describes a UML profile and metamodel for designing services. The goals of SoaML are to support the activities of service modelling and design and to fit into an overall model-driven development approach. The SoaML profile supports the range of modelling requirements for service-oriented architectures, including the specification of systems of services, the specification of individual service interfaces, and the specification of service implementations.

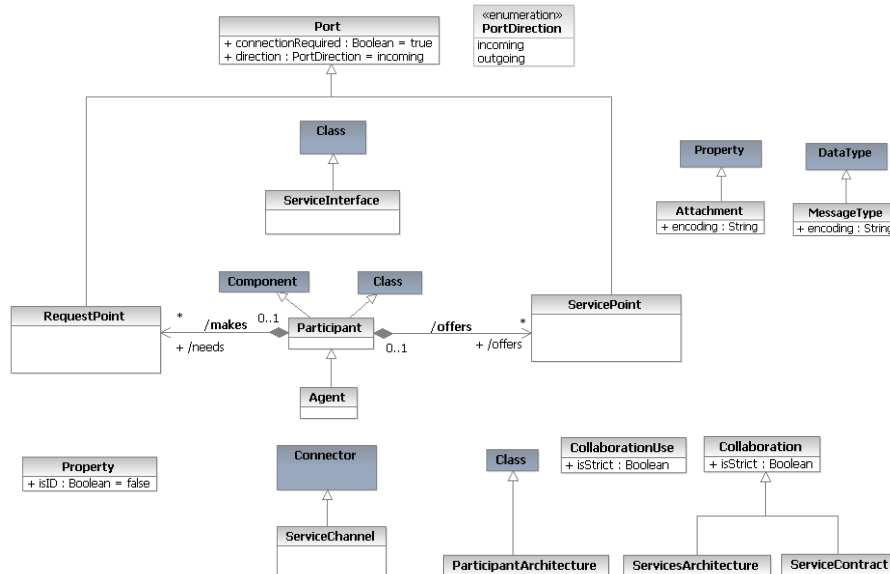


Figure 5: The SoaML profile

The SoaML profile (see Figure 4) [9] extends the UML2 metamodel to support an explicit service modelling in distributed environments. This extension aims to support different service modelling scenarios such as single service description, service-oriented architecture modelling, or service contract definition. The main extension areas are:

- **Participants** to define the service providers and consumers in a system. A Participant may play the role of service provider, consumer or both. When a participant acts as a provider it contains ServicePoints, and when a participant acts as a consumer it contains RequestPoints.
- **Service interfaces** to describe the operation provided and required to complete the functionality of a service. A ServiceInterface can be used as the protocol for a ServicePoint or a RequestPoint.
- **Service contracts** to describe interaction patterns between service entities. A ServicesContract is used to model an agreement between two or more parties. Each service role in a ServiceContract has a ServiceInterface type that usually represents a provider or consumer.
- **Service data** to describe service messages and message attachments. The MessageType is used to specify the information exchanged between services, attached to rather than contained in the message.
- **Services architectures** to define how a set of participants works together for some purpose by providing and using services. A ServicesArchitecture describes how participants work together by providing and using services expressed as ServiceContracts.

Acronyms

Acronym	Description
ATL	ATLAS Transformation Language
BMM	Business Motivation Metamodel
BPDM	Business Process Definition Metamodel
BPEL	Business Process Execution Language
BPMN	Business Process Modelling Notation
DSL	Domain Specific Language
EMF	Eclipse Modelling Framework
EPC	Event Driven Process Chain
GMF	Graphical Modeling Framework
MAS	Multiagent Systems
MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
OWL	Web Ontology Language
SOAML	SOA Modelling Language
UML	Unified Modelling Language
UPMSHA	UML Profile for Modelling for Semantically-enabled service Architectures
WSA	Web Service Architecture
WSDL	Web Service Definition Language
XSD	XML Schema Definition

References

- [1] Hahn, Christian (2009-01-14): SHAPE – Deliverable 5.1: Model transformation and deployment architecture description. http://www.shape-project.eu/wp-content/uploads/2009/01/shape_d51.pdf
- [2] Object Management Group: Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS). <http://www.omg.org/docs/ad/08-08-04.pdf>
- [3] SHAPE, "Annex I - "Description of Work"", SHAPE STREP, 15 October 2007.
- [4] Hahn, Christian (2009-12-7): SHAPE – Deliverable 5.5: Model transformations and deployment – from UPMSHA to WSA, agents, P2P, grid and SWS platforms – Final version
- [5] Kleppe, A.,Warmer, J. and Bast,W. (2003). MDA Explained, The Model-Driven Architecture: Practice and Promise, Addison Wesley.
- [6] Hahn, Christian (2009-01-14): SHAPE – Deliverable D5.2 - Model transformations from business models to UPMSHA – Final version. http://www.shape-project.eu/wp-content/uploads/2009/01/shape_d52.pdf
- [7] Hahn, C., Dmytro, P. and Fischer K.: A Model-Driven Approach to Close the Gap between Business Requirements and Agent-Based Execution. In Proceedings of the 4th Workshop on Agent-based Technologies and applications for enterprise interoperability (ATOP 2010) held in conjunction with AAMAS 2010, Toronto, Canada, June, 10th, 2010.
- [8] Hahn, Christian (2009-12-7): SHAPE – Deliverable 5.5: Model transformations and deployment – from UPMSHA to WSA, agents, P2P, grid and SWS platforms – Final version
- [9] Benguria, Gorka (2010-05-31): SHAPE – Deliverable 3.6: Integrated set of metamodels for SHA (M30).
- [10] ARIS Architecture and Reference Models for Business Process Management, August-Wilhelm Scheer, Markus Nüttgens. 2000, Berlin. Available in: http://www.wiso.uni-hamburg.de/fileadmin/WISO_FS_WI/EPKCommunity/LNCS_Geschaeftsprozessarchitektur.pdf
- [11] Business Process Modeling Notation (BPMN), OMG Document Version 1.2 2009-01-03. Available in: <http://www.omg.org/docs/formal/09-01-03.pdf>
- [12] SHAPE project Website: <http://www.shape-project.eu/>