



D11.4
MDA/MDI Model Transformation:
Application to MDSEA
M15 issue

Document Owner:	Ricardo GONCALVES (Uninova)
Contributors:	Carlos Agostinho (Uninova), Edgar Silva (Uninova), Yves Ducq (UB1), Hassan Bazoun (Hardis), Hadrien Boyé (Hardis)
Dissemination:	Public
Contributing to:	WP 11
Date:	22.02.2013
Revision:	Version 3.0

VERSION HISTORY

1	DATE	NOTES AND COMMENTS
2	17.12.2012	CREATION OF THE TABLE OF CONTENTS
3	31.01.2013	FIRST REVIEW OF INPUT FROM D11.3 AND MODIFICATION TABLE OF CONTENTS CONSIDERING THE REVIEW REPORT
4	04.02.2013	FIRST INPUT BY UNINOVA (CHAPTER 4) AND UB1/HARDIS (BSM-TIM TRANSFORMATIONS)
5	08.02.2013	FIRST INTEGRATED DRAFT (VERSION 1.0)
6	15.02.2013	INPUT TO CHAPTERS 5, 6 AND 7
7	18.02.2013	TOC REVISION AND SECOND INTEGRATED DRAFT (CIRCULATED ALSO FOR PEER-REVIEW) (VERSION 2.0)
8	19.02.2013	MAPPING FROM EA* TO UML INCLUDED
9	21.02.2013	CORRECTION ACCORDING TO PEER-REVIEW COMMENTS AND COMPLETION OF “RECOMMENDED PLAN FOR IMPLEMENTATION IN THE SLM TOOLBOX”
10	22.02.2013	FINAL VERSION 3.0

DELIVERABLE PEER REVIEW SUMMARY

ID	Comments	Addressed (✓) Answered (A)
1	The official notation of workpackages should be used (p. 6 and 7)	✓
2	The font size used in the figure should be increased: in a paper version they could be unreadable (p.17)	✓
3	It is not possible to read the text in the diagrams. It is suggested to enlarge the pictures of section 8 putting one picture per page	✓
4	Misprints to be fixed (see revision in the document)	✓
5		
6		
7		

TABLE OF CONTENTS

1	LIST OF ACRONYMS AND ABBREVIATIONS	5
2	EXECUTIVE SUMMARY	6
3	INTRODUCTION	7
4	MDSEA TRANSFORMATION ARCHITECTURE AND OTHER PREVIOUS DEVELOPMENTS	8
4.1	MSEE METHOD	8
4.2	MODEL DRIVEN SERVICE ENGINEERING (MDSEA) ARCHITECTURE	8
4.3	TRANSFORMATIONS FRAMEWORK AND ARCHITECTURE	9
4.3.1	<i>Ontologies to Support Mappings and Transformations</i>	10
4.4	EXISTING MSEE ONTOLOGIES	10
4.5	CONCLUSIONS AND CONSIDERATIONS	10
5	SPECIFICATION OF MDSEA MODEL TRANSFORMATIONS	12
5.1	INTEGRATED METHODOLOGY FOR MODEL TRANSFORMATIONS WITHIN MDSEA	12
5.1.1	<i>Step 1: Formalize the Meta-Models</i>	12
5.1.2	<i>Step 2: Build the MSEE Reference Ontology</i>	13
5.1.3	<i>Step 3: Define the Model Mappings</i>	14
5.1.4	<i>Step 4: Implement the Model Mappings</i>	15
5.1.5	<i>Step 5: Execute Transformations</i>	15
5.2	USE OF ONTOLOGIES TO SUPPORT TRANSFORMATIONS	16
5.2.1	<i>Rationale for Ontologies</i>	16
5.2.2	<i>Ontologies in the Transformation Process</i>	16
5.3	VERTICAL TRANSFORMATIONS: MODEL MAPPINGS ALONG THE MDSEA AXIS	18
5.3.1	<i>Mapping from BSM level to TIM</i>	19
5.3.2	<i>BSM-TIM: Mapping from Extended Actigram* (EA*) to BPMN</i>	19
5.3.3	<i>BSM-TIM: Mapping from EA* to UML</i>	22
5.3.4	<i>Mapping from TIM level to TSM</i>	24
5.3.5	<i>TIM-TSM: Mapping from BPMN 2.0 to WS-BPEL 2.0</i>	25
5.3.6	<i>TIM-TSM: Mapping from BPMN 2.0 to WSDL 2.0</i>	25
5.4	HORIZONTAL TRANSFORMATIONS ACROSS THE ECOSYSTEM AXIS	26
5.4.1	<i>USDL interoperability: Mapping from EA* to USDL</i>	27
6	RECOMMENDATION FOR TRANSFORMATIONS IMPLEMENTATION IN THE SLM TOOLBOX	28
6.1	SPECIFICATION FOR THE TRANSFORMATIONS METHODOLOGY IMPLEMENTATION	28
6.1.1	<i>Actor Interaction Requirements</i>	28
6.1.2	<i>Functional Specification</i>	28
6.1.3	<i>Detailed Technical Specification</i>	28
6.2	HORIZONTAL TRANSFORMATIONS ACROSS THE ECOSYSTEM: HOW TO INCLUDE IN THE SLM TOOLBOX	31
6.3	RECOMMENDED PLAN FOR IMPLEMENTATION IN THE SLM TOOLBOX	32
7	SPECIFIC IMPLEMENTATIONS	34
7.1	VERTICAL TRANSFORMATIONS (BSM TO TIM)	35
7.1.1	<i>Core Concepts</i>	35
7.1.2	<i>Transformation from EA* to BPMN 2.0 (Vertical – language level)</i>	35
7.2	VERTICAL TRANSFORMATIONS (TIM TO TSM)	35
7.2.1	<i>Transformation from BPMN to WS-BPEL (Vertical – language level)</i>	35
7.2.2	<i>Transformation from BPMN to WSDL (Vertical – language level)</i>	36
8	INTEGRATED SLM TOOLBOX TRANSFORMATION EXAMPLE FROM THE BIVOLINO USE-CASE	38
9	CONCLUSIONS	45
10	REFERENCES	46
	ANNEX A – MDSEA CORE CONCEPT META-MODELS	47
	BSM META-MODEL	47
	TIM META-MODEL	49
	TSM META-MODEL	50
	ANNEX B – SELECTED LANGUAGES META-MODELS	51
	EXTENDED ACTIGRAM STAR (EA*)	51

UML 2.0 STATECHARTS (ALSO KNOWN AS STATE MACHINE)	54
BPMN 2.0	55
WS-BPEL 2.0	56
WSDL 2.0	58
USDL	59

LIST OF FIGURES

Figure 1: Methodology for servitization system definition and implementation.....	8
Figure 2: Model Driven Service Eng. Architecture	8
Figure 3: Revised MDSEA Transformations Framework (including Architecture)	9
Figure 4: Macro steps in methodology for transformations within MDSEA	12
Figure 6: Activities towards the formalization of the source and target meta-models (deliverable D11.3).....	13
Figure 7: Activities towards the MSEE reference ontology building and extension	14
Figure 8: Activities towards transformation execution	15
Figure 9: Activities towards the usage of the MSEE reference ontology	17
Figure 10: EA* to UML transformations	23
Figure 11: Example of a horizontal transformation process at BSM level	27
Figure 13: Detailed Transformation	30
Figure 14: Transformations architecture used in implementations	34
Figure 15: Simple BSM representation of a shoes customization service	35
Figure 16: Result of the transformation of the BSM model in Figure 15	35
Figure 17: BPMN 2.0 to WSDL 2.0 Transformation Example.....	37
Figure 18: Bivolino modeling and transformation scenario (see future deliverable D15.3 for more detail).....	38
Figure 19: Sequence of snapshots illustrating the Bivolino modeling and transformation scenario (see future deliverable D15.3 for more detail)	44
Figure 20: BSM Core Concepts Meta-Model.....	48
Figure 21: TIM Core Concepts Meta-Model.....	49
Figure 22: TSM Core Concepts Meta-Model.....	50
Figure 23: BaseElement	51
Figure 24: EA* Conceptual Meta-Model	52
Figure 25: UML 2.0 STATECHARTS Meta-Model	54
Figure 26: WS-BPEL Meta-Model (from ebPLM(2007))	57
Figure 27: WSDL 2.0 Meta-Model	59

LIST OF TABLES

Table 1: BSM-TIM mapping table (Deliverable D11.3).....	19
Table 2: EA* to BPMN2.0 mapping table (revised version)	19
Table 3: EA* to UML2.0 State Charts mapping table	23
Table 4: TIM-TSM mapping table	24
Table 5: BPMN2.0 to WSDL2.0 mapping table	25
Table 6: Plan for modelling/ontology support functionalities.....	32
Table 7: Recommended plan for transformation functionalities.....	33
Table 8: Flow constraints	54
Table 9: Objects in WSDL 1.1 / WSDL 2.0.....	58

1 List of Acronyms and Abbreviations

ATL	Atlas Transformation Language
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
BSM	Business Service Models
EA*	Extended Actigram Star
EMF	Eclipse Modeling Framework
IAO	Intangible Assets Ontology
LCIM	Levels of Conceptual Interoperability Model
MDA	Model-Driven Architecture
MDI	Model-Driven Interoperability
MDSEA	Model-Driven Service Engineering Architecture
MOF	Meta Object Facility
MSEE	Manufacturing Services Ecosystem
OCL	Object Constraint Language
OMG	Object Management Group
OWL	Ontology Web Language
QVT	Query View Transformation
SLM	Service Lifecycle Management
SOAP	Simple Object Access Protocol
TAO	Tangible Assets Ontology
TIM	Technical/Technology Independent Models
TSM	Technical/Technology Specific Models
UML	Unified Modeling Language
USDL	Unified Service Description Language
VME	Virtual Manufacturing Enterprise
WSDL	Web Service Definition Language
XMI	XML Metadata Interchange format
XSLT	eXtensible Stylesheet Language Transformations

2 Executive Summary

This deliverable follows the work of D11.3 in the scope of WP11. It is specific on MDA/MDI model transformation and envisages to complement the MDSEA architecture with a transformations methodology and framework. Hence, the content here included must be considered as a complementary release (M15) of deliverable D11.3 (M8) and as a presentation of the final specification of the MDA/MDI model transformation methodology in the frame of MDSEA.

The main added materials of this release are:

- **the integration of the work in a comprehensive transformations methodology,**
- **the improvement of the specifications on how to use ontologies in the frame of MDSEA**
- **the specification of concrete mappings to be implemented**
- **detailed recommendation and plan for the implementation in the SLM Toolbox**
- **results of MDSEA transformations application to a specific case study.**

The MSEE servitization methodology begins at the strategic level of companies, where depending on their objectives, modelling and vertical transformations are required to go from the desired strategy, a “to-be” business specific model (BSM), towards a detailed functional definition (TIM) and practical specification (TSM). Several models have been chosen at each modelling level and transformations identified as required to go from one level to another towards service system implementation, or at the same level to enable sharing and interoperability among different enterprises (horizontal transformations). In this context, as reported in deliverable D11.3, the MDSEA transformations framework is defined along three axes:

- Modelling levels, defined according to the reference modelling architecture categorization proposed by OMG, which envisages that real world data is modelled using 4 levels that go from data itself (M0) to the meta-meta-model (M3);
- MDSEA levels, which, being inspired on the MDA/MDI enables Service System modelling around 3 abstraction levels, i.e. BSM, TIM, and TSM;
- Ecosystem integration, which, starting from a minimum of 2 systems represents the P2P integration among the multitude of service systems part of the enterprise ecosystem.

Considering that simple type mappings are generally insufficient to specify a complete and fully automatic transformation, the transformations architecture proposed includes semantic knowledge to help in the dynamicity and automation of the process. The integrated methodology for model transformations within MDSEA follows preparatory steps (“Formalize the Meta-Models” and “Build the MSEE Reference Ontology”), steps for the design of the transformation (“Define the Model Mappings” and “Implement the Model Mappings”) and one final step for the execution of the transformation.

Important milestones for this deliverable version are to specify the use of ontologies along the transformations process, as well as to provide concrete mappings, demonstrate their implementation, and detail a recommendation and plan for the methodology implementation in the SLM Toolbox. Complementing D11.3 that was focused on transformations from BSM to TIM, this document demonstrates the feasibility of Technical Independent Models (TIM) to Technical Specific Models (TSM) transformations. Hence, specific mappings and transformations examples are included.

3 Introduction

The objective of this document is to report advances and present a consolidated view on the work being developed in MSEE in the domain of model transformation. Indeed, the content of deliverable D11.4 must be considered as a complementary release of deliverable D11.3 and a presentation of the final specification of the MDA/MDI model transformation methodology in the frame of MDSEA. The work is part of work package 11 that is specifying the Modelling platform that will automate as much as possible the models transformation of from conceptual level to more technical levels, thus contributing with direct specifications to WP15.

Based on the MDSEA architecture, defined in the deliverables D11.1 and D11.2, several models have been chosen at each modelling level and transformations identified and required to go from one level to another towards service system implementation. The idea is to provide the capability to transform a business specific model into a functional one, and that into a technology specific model envisaging the generation of concrete software and services. The capability to harmonize models specified by different enterprises, enabling interoperability and collaboration (e.g. process orchestration, service matching) within the ecosystem must also be addressed.

In the first part of this document (section 4), the main developments so far will be reminded, clarifying the transformation requirements derived from the MSEE method, and the MDSEA architecture. The first issue of this document presented an abstract transformations architecture integrated in a 3 axis-framework and focused implementations on BSM to TIM static vertical transformations. In section 5, the previous research developments are complemented and integrated into a methodology for model transformations within MDSEA. This provides concrete specifications on the steps to be taken from the transformation preparation until its execution. In this chapter, the use of ontologies to raise the transformations completeness, dynamism, and automation levels, supporting transformations at both vertical (MDSEA axis of the transformations framework) and horizontal levels (ecosystem axis) is explained and extrapolated to complement existing implementations.

Section 6 is practical, deriving concrete recommendations for the SML Toolbox implementation. Beginning from an analysis of user interactivity, this section translates the transformations methodology into detailed technical specifications. A plan for implementation in the SLM Toolbox is included, clearly distinguishing what was done at the time of D11.3 (M8), what is currently being done at the time of D11.4 (M15), what will be done at the time of the review (M18), at the time of the conclusion of the toolbox (M24), and towards the future (outside MSEE).

Specific implementations are reported in section 7, and a concrete example from the Bivolino use-case is included in section 8, highlighting how the transformations are needed on a real scenario, already being integrated on on-going implementations of the SLM Toolbox.

At the end of this document, after a short conclusion, the future work to be done and presented in the next deliverables of WP11 will also be explained.

4 MDSEA Transformation Architecture and other Previous Developments

In order to contextualize this deliverable, it is important to wrap-up the previous developments concerning MSEE model transformations. Illustrative figures and references to other MSEE documents are used to avoid repetition.

4.1 MSEE Method

Deliverable D11.1 “Service concepts, models and method: Model Driven Service Engineering”, followed by deliverable D11.3 “MDA/MDI Model Transformation: Application to MDSEA” has specified and improved the description of the MSEE method for enterprises that want to evolve towards service-oriented business methods (servitization).

The method begins at the strategic level of companies, where depending on their objectives, modelling and vertical transformations are required to go from the desired strategy, a “to-be” business specific model towards a detailed functional definition and practical implementation (Figure 1).

Since models can be shared to enable collaboration among different companies (e.g. process orchestration) operating at several domains and using different technologies, horizontal transformations are also considered to ensure interoperability.

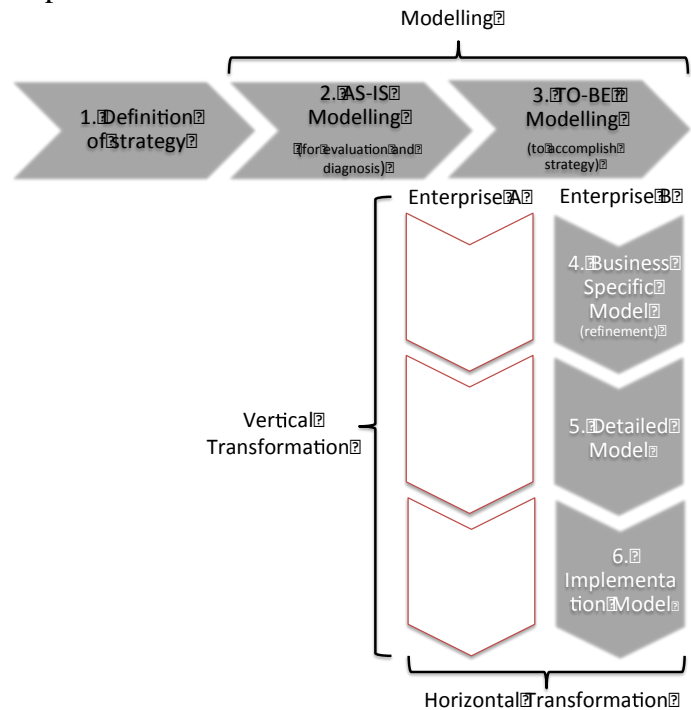


Figure 1: Methodology for servitization system definition and implementation

4.2 Model Driven Service Engineering (MDSEA) Architecture

Based on the above principles, the MDSEA architecture was defined in deliverable D11.1 (and its latest issue, D11.2) to model and guide the vertical transformation from the business requirements of the enterprise into detailed specifications of components that must be implemented to support the servitization process. In this architecture (see Figure 2), several modelling levels are defined to have a progressive specification of service system components at the business (Business Service Modelling - BSM), functional (Technology Independent Modelling - TIM), and technological (Technology Specific Modelling - TSM) levels. Taking into account the technology, MDSEA models integrate the requirements leading to the implementation of a solution in IT, organization or physical domains.

The approach implies that the different levels should use dedicated service modelling languages that represent the system with the appropriate level of description. GRAI Integrated Modelling has been considered as a reference for the BSM level, but further detail on the analysis and

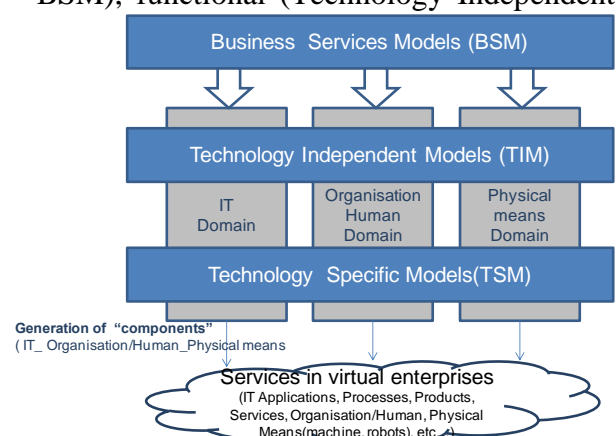


Figure 2: Model Driven Service Eng. Architecture

selection of the most appropriate languages can be found in deliverable D11.2 “Service concepts, models and method: Model Driven Service Engineering (M12 issue)”.

4.3 Transformations Framework and Architecture

As explained, the MSEE method applies the distinction between vertical and horizontal transformations, providing interoperability and portability at the same degree of relevance as the traceability features, linking requirements, design, analysis, and testing models of the several MDSEA abstraction levels. In this context, deliverable D11.3 “MDA/MDI Model Transformation: Application to MDSEA”, defined a framework for MDSEA transformations along 3 different axis (see Figure 3). It envisages a formal specification of models according to the OMG (OMG 2011a) categorization to enable vertical transformations from BSM to TSM as well as horizontal ones to integrate different service systems.

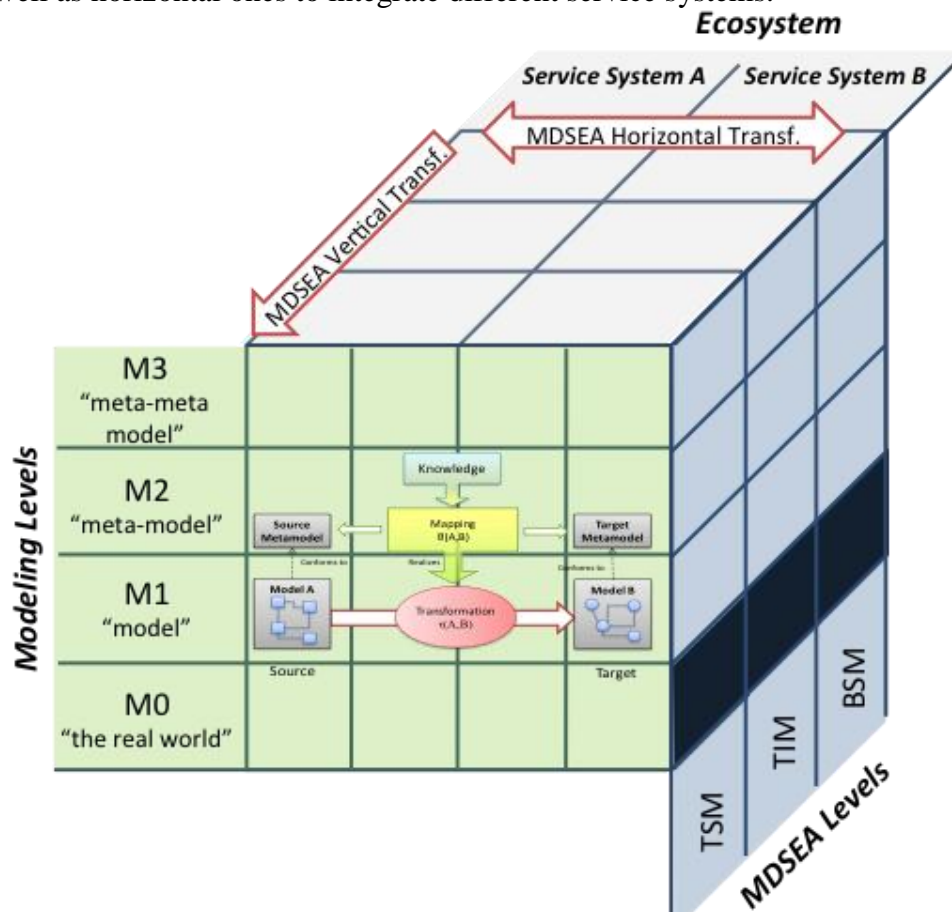


Figure 3: Revised MDSEA Transformations Framework (including Architecture)

Has specified, when performing a model transformation (from ModelA to ModelB), one is converting instances of a source model to instances of another model, the target, and an explicit or an implicit mapping at the same meta-modelling level has to be performed. Thus, as depicted in the generic transformations architecture included in the framework (frontal pane – green highlight), the idea is that when performing a transformation “ $\tau(A,B)$ ” at a certain level “ i ”, this transformation has (implicitly or explicitly) to be designed by taking into account mappings “ $\theta(A,B)$ ” at level “ $i+1$ ”. Once the “ $i+1$ ” level mapping is complete, executable languages (e.g. ATL¹) can be used to implement the transformation itself.

¹ ATL – Atlas Transformation Language (www.eclipse.org/m2m/atl/)

4.3.1 Ontologies to Support Mappings and Transformations

Transformations are traditionally static processes that once defined can be repeated any number of times achieving the same results. The major difficulty resides on the definition of the mapping, which typically needs to involve specialists on the source and target models, as well as programmers to implement it in a specific language such as ATL. The problem comes in the case of dynamic environments that can demand for a frequent reformulation of mappings. Enterprise ecosystems fall under this category since they are open environments that enable different enterprises to enter and leave according to their business objectives.

Considering that, and mostly focused on the need to support interoperability (horizontal transformations), Deliverables D11.3 and then D11.5 “MSEE architecture for Service Modelling” suggest that explicit knowledge represented through dedicated ontologies can help to achieve the desired dynamicity and automation. Annotation to reference domain and technical ontologies can be done during the modelling process, and used to support a semi-automatic generation of the mapping, namely service matching to enable interoperability.

The vertical transformation along the MDSEA axis can also contribute and benefit (depending on specific cases) to and from this ontological support.

4.4 Existing MSEE Ontologies

Currently there are two ontologies/taxonomies formally defined in the scope of MSEE:

- One on intangible assets - **TAO** (WP22 – see deliverable D22.3 “Development of methods for virtualization of MSEE intangibles”) and;
- Another one for tangible assets – **IAO** (WP23 – see deliverable D23.3 “OMSE Management Framework for Tangible Assets”).

TAO has been designed to virtualize real-world tangible assets and meet the requirements of MSE to *handle*, *share* and *communicate* as well as *combine* tangible assets in ecosystems. It provides the means for merging and aligning semantic models of two or more tangible assets and serves as semantic and structural means for effective and efficient ICT-driven handling of tangibles. IAO has a similar purpose for intangibles, organizing them under *Human*, *structural* and *relational* capital.

Both ontologies are not exhaustive with respect to their number of concepts or instances, but they prove that real-world in-/tangible assets in Ecosystems or VME (Virtual Manufacturing Enterprise) can be modelled, formalized, not to say virtualized by means of these ontologies and thus represented as in-/tangible assets as a service (I/TaaS). These services are then published in a USDL repository, so that Ecosystem members can use and combine them towards marketable (manufacturing) products/services.

Currently, there is a plan to converge both ontologies and further populate them.

4.5 Conclusions and Considerations

After the specification of the MSEE method, which contributed to the identification of concrete transformation requirements, the MDSEA architecture provided the building blocks for VME service development, scoping the work to be implemented in MSEE. The high level requirements are:

- The capability to transform a business specific model into a functional one that can then be complemented by a system architect;

- The capability to transform a functional model into a technology specific one envisaging the generation of concrete software and services;
- The capability to harmonize models specified by different enterprises, enabling interoperability and collaboration (e.g. process orchestration, service matching) within the ecosystem.

Answering to those requirements, an abstract transformations architecture and framework has been formalized. Until now, deliverable D11.3 focused on BSM to TIM vertical transformations (first bullet). Nevertheless the architecture proposed enables to do more, e.g. the inclusion of semantic support mechanisms such as ontologies to raise the transformations completeness and automation levels.

In this line, MSEE has so far provided two ontologies on tangible and intangible assets that should be converged and made part of a larger reference MSEE ontology enabled in the form of XaaS. Hence, becoming capable of supporting not only the virtualization of real-world and intangible assets but also the semantic enrichment of models at the IT, physical and human-related domains of MDSEA.

5 Specification of MDSEA Model Transformations

Based on past research initiatives, e.g. MDI, the previous issue of this deliverable (i.e. D11.3) has reported the advances in the specification of a transformations framework for service systems (see Figure 3). In this section, that work is continued further detailing and specifying the methodology to enable and conduct SLM transformations, and envisaging integration in the MSEE SLM Toolbox.

5.1 Integrated Methodology for Model Transformations within MDSEA

Applying the distinction between vertical and horizontal transformations, the MDSEA transformations are specified according to parameters defined along the 3 axes of Figure 3. Considering that simple type mappings are generally insufficient to specify a complete and fully automatic transformation, the transformations architecture has adapted the OMG MDA Guide (OMG 2003) to include semantic knowledge to help in the automation of the process. This way, as illustrated by the diagram of Figure 4 and described in the following sub-sections, the recommended methodology for transformations within MDSEA follows:

- 2 preparatory steps – “Formalize the Meta-Models” and “Build the MSEE Reference Ontology”;
- 2 other for the design of the transformation – “Define the Model Mappings” and “Implement the Model Mappings”;
- 1 for its execution – “Execute Transformations”.

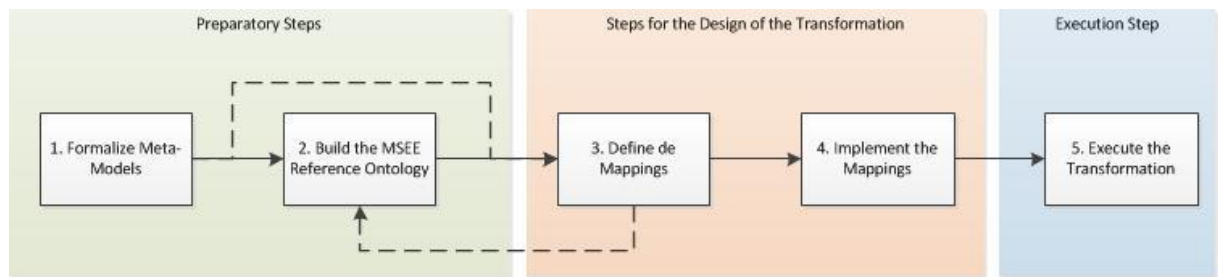


Figure 4: Macro steps in methodology for transformations within MDSEA

5.1.1 Step 1: Formalize the Meta-Models

The first step of the methodology, and a mandatory pre-requisite, is the formalization of the source and target MDSEA meta-models.

In this line of work, the templates defined on deliverables D11.1 and D11.2 provide a “zoom” view on the BSM, TIM and TSM core constructs meta-models, separating each MDSEA concern (e.g. zoom on service, process, decision, etc.). As explained along deliverable D11.3 and

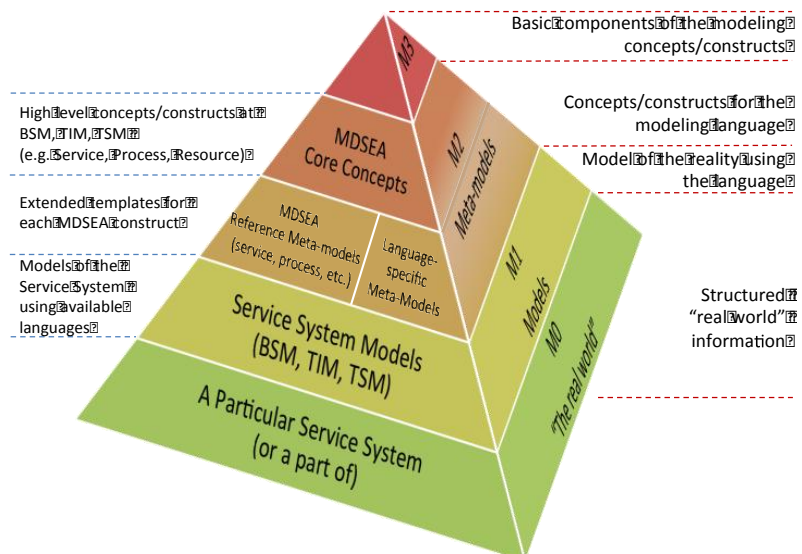


Figure 5: MDSEA mapped to the OMG 4 level architecture (Deliverable D11.3)

illustrated by Figure 5 and Figure 6, they need to be extended with the concepts that each language provides in addition (e.g. Extended Actigram* has more details than the one envisaged in the BSM “Process” templates). Together, they fulfil the first step, creating the M2 “Meta-model” level of the OMG architecture (OMG 2011a) and becoming the basis for the mapping definition, envisaging alignment at the M1 “Model” level (step 3) when instantiated by specific BSM, TIM and TSM models.

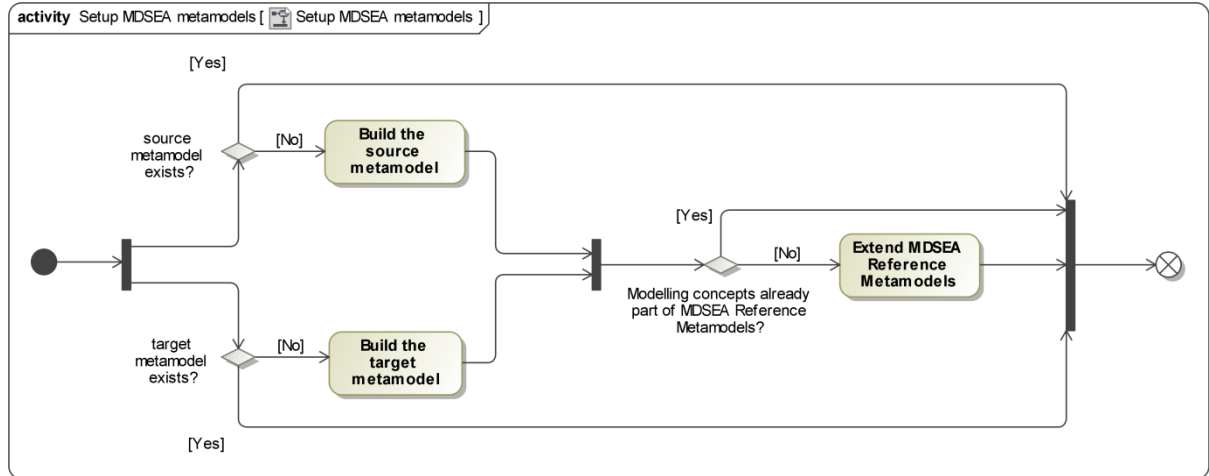


Figure 6: Activities towards the formalization of the source and target meta-models (deliverable D11.3)

5.1.2 Step 2: Build the MSEE Reference Ontology

Transformations need syntactic alignment given by the meta-models as a pre-requisite, so that the approach for processing the information will be interpretable from a known structure. However, once the syntactical correctness has been verified, semantic interpretation, which goes beyond syntax or structure, must be understood and unambiguously defined to enable an increased automation based on the context of the mapping definition. With the use of ontologies to support the transformations, MSEE gains the following capabilities:

- **Searching/Analysing** – support to modelling process and common understanding on all levels through an *MSEE Reference Ontology*;
- **Traceability** – historical tracking of mappings and transformations using a *Mapping Knowledge Repository* (can be integrated in the reference ontology or not);
- **Reasoning** – automation (*MSEE Reference Ontology and Mapping Knowledge Repository*), namely in the generation of the transformations code and validation of mappings.

Nonetheless, it is of public domain that ontologies building is a slow procedure that may take a considerable time. For this reason MSEE envisages a semi-automatic construction of the ontology, based on the MDSEA meta-models, complemented with user knowledge collected along to modelling process. Still, not to prevent real demonstration of the transformations process in the MSEE frame, the transformations methodology specified enables to jump over this step directly from step 1 to step 3.

Figure 7 details the activities towards the MSEE reference ontology building and extension:

- Initially, the structure of the meta-models (either at BSM and TIM levels) can be automatically extracted and used in the generation of the ontology core structure;
- Then, existing ontologies such as the MSEE tangible and intangible assets ontologies can be used to extend that initial structure. In this particular case, manual links can be defined among the assets and the concept of “resource” or “process”;

- Another important activity towards the ontology creation is the usage of the modelling process, i.e. the instantiation of the meta-models with specific use-cases, to extract domain knowledge.
- This domain information can also extend the reference ontology, complementing it with domain-specific knowledge concerning the 4 use-cases of MSEE.

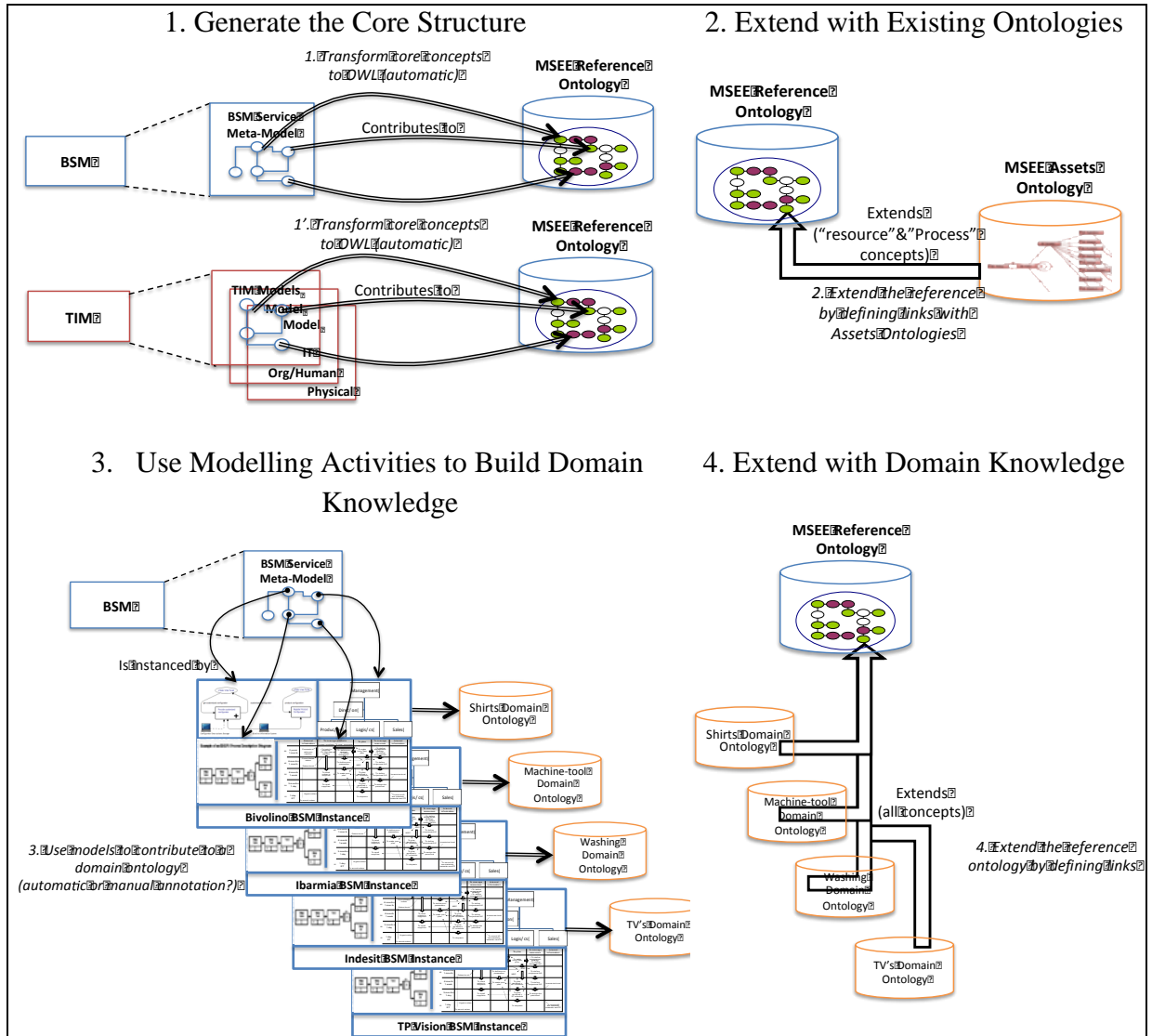


Figure 7: Activities towards the MSEE reference ontology building and extension

5.1.3 Step 3: Define the Model Mappings

Step 3 of the transformations methodology marks the beginning of the actual design of the transformations by defining the model mappings that relate the concepts of each meta-model. If existing, the ontology can contribute to the mappings definition since it adds a common understanding of certain concepts used in the meta-models. Inversely, the mapping process can also contribute to the ontology.

From a syntactic point of view, the mapping is a morphism (<http://en.wikipedia.org/wiki/Morphism>) that must ensure the consistency of source and target models, and is created relating each element of the source with a correspondent element in the target (1-to-1, 1-to-n, or m-to-n) while leaving both intact. In transformations, the source model is transformed using a function that applies a mapping to produce a different target model. This function can be expressed either *explicitly*, using graphs, sets, tuples, or even

mapping tables relating multiple or single constructs and stored in a physical location; or *implicitly* in the developer's mind. However, in both cases is necessary to implement them using a transformation language (step 4).

5.1.4 Step 4: Implement the Model Mappings

Model transformation is an important activity in Model-Driven Engineering, and OMG recognized this by issuing the Query/Views/Transformations (QVT) request for proposals to seek an answer compatible with its MDA standard suite. Many contributions were submitted which led to several transformation languages with support for automatic model transformation execution. Some of these are based on the Object Constraint Language (OCL), like QVT itself and ATL, which despite not being a standard is one of the most used, having a large user's base. Nevertheless, as enumerated in Czarnecki & Helsen (2006), others can also be used and applied to the implementation of the model mappings, e.g. Xtend/Xpand, UMLX, AToM3, MTL, etc. (see deliverable D11.1 and D11.2).

As analysed by Agostinho (2011), some of the above languages are ideal for the representation structural mappings, others for semantic maps, providing good human traceability, while others are more formal and mathematical based. However, none provides the capability or the APIs to translate explicit mappings into executable code. Mappings implemented with them are normally static and any change obliges to manually rewrite code.

Benefiting from a good JAVA integration that enables to address the above problems in the future and having a considerable amount of support through online communities, ATL has been the elected language for MDSEA mappings implementation.

5.1.5 Step 5: Execute Transformations

Following the architecture and methodology specified, once the mappings (together with the MSEE reference ontology) are defined and implemented (see Figure 8 - from left to right), automatic transformations between the source and target models can be achieved (right side of Figure 8 – from top to bottom). Once the mapping among the source and target models has been previously defined, one only needs to load and implement it as explained in step 4.

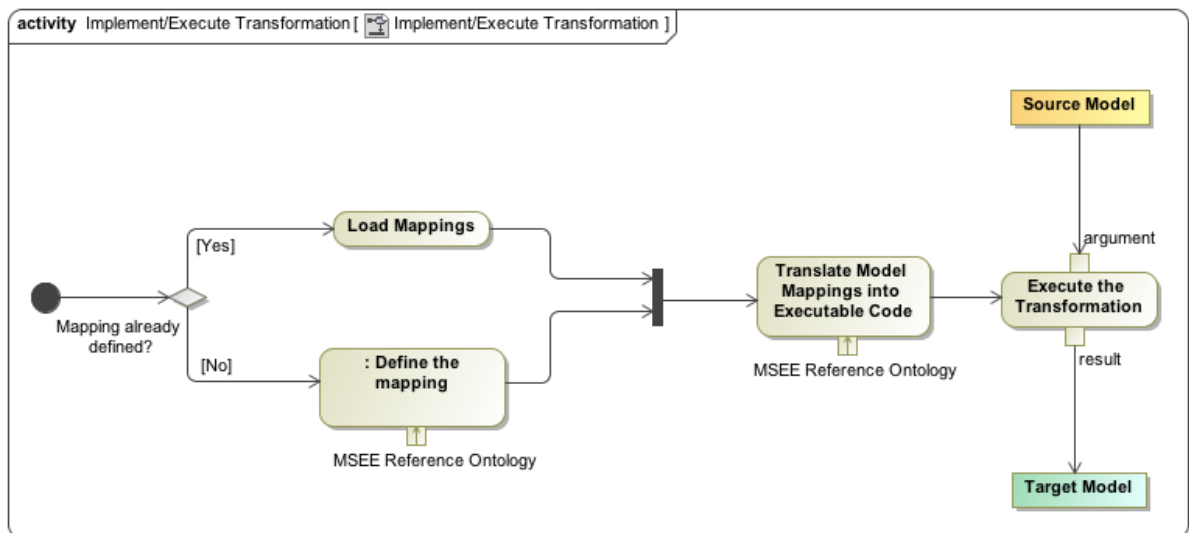


Figure 8: Activities towards transformation execution

5.2 Use of Ontologies to Support Transformations

5.2.1 Rationale for Ontologies

As explained in the previous deliverables and summarized in section 4.3.1, transformations are traditionally static processes that once implemented based on the mappings defined can be repeated any number of times achieving the same results. However, it can be argued if just by using conceptual models, semantics are being given a proper attention and formalization. They are recognized by the research community as an important area for models alignment and one of the levels of interoperability to consider within an enterprise (Athena IP 2006).

Wenguang Wang et al. (2009) place semantic interoperability in the middle of the LCIM (levels of conceptual interoperability model) scale towards a seamless conceptual interoperability, where more than an agreement between all systems on a set of terms, companies need a shared understanding of the system's conceptual models. The major difficulty resides on the definition of the mapping, which typically needs to involve specialists on the source and target models as well as some transformation specialists. Some questions might help understanding the usefulness of ontologies:

- ***Is possible to automate the definition of mappings?*** – The process involved in the definition of the model mappings is knowledge intensive and requires an extensive understanding of the models as well as their context (e.g. domain of application). Normally it is manual process, but in some cases mappings could be semi-automatic, generated based on knowledge bases that trace association between concepts.
- ***Are the models self-explanatory?*** – Typically models are analysed under the form of diagrams to empower the Human understanding, but due to graphical and space constraints, the information annexed to each modelling concept is implicit (each symbol means a different thing), and the model itself depends on the context as well as the personal preferences to the one who defined it. Hence its analysis might not be straightforward.
- ***What happens if the modeller needs to change along the service (re)engineering?*** – Related with the previous question, if the modelling process is collaborative and for some reason the person responsible for a specific part or version of the model becomes unavailable, his understanding of the model might not be easily transmitted.
- ***How to verify/change a specific implementation of a transformation?*** – When there is not a dynamic association among the mappings defined at step 3 and step 4 of the transformations methodology, it is not possible to do so without having to navigate through code. This is a very time consuming activity and requires the participation of programmers and technicians.

For these reasons it would be important to have ontologies for a continuous explicit representation of semantics associated to the models, domains and the mappings (see section 5.1.2 for details on the recommended ontology building procedure)

5.2.2 Ontologies in the Transformation Process

In practice, ontologies work as repositories for storing knowledge in a parseable structure. They can support automation, by enabling intelligent services to work on top of them and providing the Human actor additional semantics throughout a certain activity.

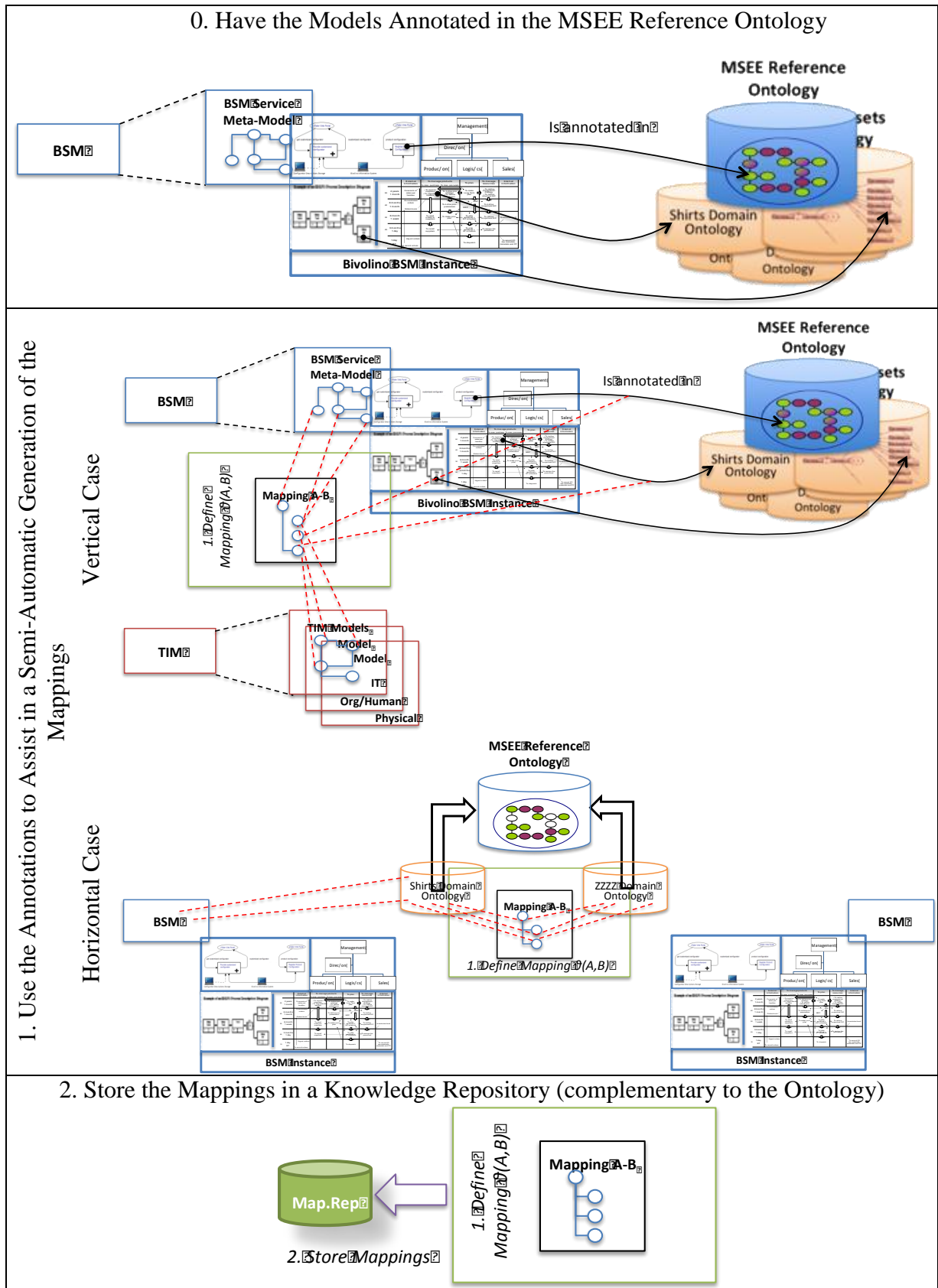


Figure 9: Activities towards the usage of the MSEE reference ontology

As illustrated in Figure 9, and explained along section 5.1.2 “Step 2: Build the MSEE Reference Ontology”, it is important that during the modelling process, BSM/TIM/TSM models are annotated in the MSEE Reference Ontology. Having that, ontologies can be used in both vertical and horizontal transformation processes for a semi-automatic generation of mappings:

- Intelligent services (e.g. agents) can use the knowledge stored, querying and relating it with a specific problem, while reasoning over it to suggest an initial set of mappings;
- The Human user validates the suggested mappings and complements them using not only its tacit knowledge, but also the explicit semantics associated to each of the annotated concepts.

Horizontal transformations derive more benefits from the use of ontologies since they are frequently associated with a need to integrate different service systems using different information structures or even domains of knowledge (case illustrated in figure). This type of transformations pose higher interoperability problems than the vertical ones that are conducted always in the same domain of knowledge and derived from the same requirements at BSM level.

As depicted in the bottom of Figure 9, in transformations (especially in horizontal ones) every mapping between models of business partners can be stored explicitly (complementary to the ontology) and accessed by their local systems. Such repository ontology for traceability of mappings would allow communities to build systems and services with reasoning capabilities over the mappings defined. They would be able to understand each other's representation format at a certain point in time, without having to change their data and schema import or export processes. The mapping repository can be seen as a meta-model for transformations, and from a technical point of view, it can also be interesting if one wants to reconstruct the transformation scripts at a later stage (because ATL is static), or outside the SML Toolbox, e.g.:

- The service models may change and one needs to redo a part of the mapping and recompile ATL, or
- ATL is evolving and in the future one may want to rewrite the scripts using new features of the language, or
- Instead of ATL one could prefer to implement transformations with other language (e.g. XSLT).

The transformations methodology depicted before, supports these needs but MSEE has not yet conducted specific research to tackle it (see Agostinho (2011) and Annex 1 of Deliverable D11.2 for more information of a possible mappings knowledge repository - Communication Mediator KB).

5.3 Vertical Transformations: Model Mappings along the MDSEA Axis

Vertical transformations that are to be implemented along the MDSEA axis of the transformations framework of Figure 3, require the definition of model mappings among:

- BSM and TIM core concept meta-models;
- Selected languages at BSM and TIM levels
- TIM and TSM core concept meta-models;
- Selected languages at TIM and TSM levels.

At the time of the deliverable, besides the mappings between each of the core concept meta-models, mappings had been defined among some of the selected languages (Deliverable D11.2): EA* and BPMN; EA* and UML; BPMN and BPEL; and BPMN and WSDL.

All meta-models relevant for the mapping definitions are available in Annex.

5.3.1 Mapping from BSM level to TIM

The following Table 1, below, shows the correspondence between the core constructs at the BSM and TIM levels. It has been initially presented in Deliverable D11.3, and it remains valid. Nevertheless it is presented again in this document due to its relevance to specific implementations of section 7.

Table 1: BSM-TIM mapping table (Deliverable D11.3)

BSM		TIM	
Service		Service	
Customer		-	
Stakeholder		-	
Partner		-	
Product		-	
Functionality		-	
Resource	Resource.type = IT	Resource	EnterpriseApplication
	Resource.type = physical mean		PhysicalMean
	Resource.type = human		Human
-		Information	
Process		Process	
Decision		-	
Organization		Organization	
-		Organization unit	
Performance Indicators		-	
Value		-	

As observed, the mapping between BSM and TIM core construct meta-models is only partial. There is at least nine constructs at BSM level which have no correspondence at the TIM level (Customer, Stakeholder, Partner, Product, Functionality, Decision, Decision Structure, Performance Indicators, Value), and must be complemented with additional modelling at TIM instances level that take these constructs into account.

This is a classic example of how the existence of an MSEE reference ontology could help. If not in a fully automatic way due to the conceptual differences among the meta-models and languages used, at least in support to the modelling activities of the architect at TIM level. The ontology can register the knowledge that is not transformed, giving suggestions and semantically meaningful hints at the time of TIM modelling.

5.3.2 BSM-TIM: Mapping from Extended Actigram* (EA*) to BPMN

The mapping of concepts proposed for the transformation creates correspondences and links between concepts and their relations from EA* to BPMN language. It is a translation of constructs and their relations from one meta-model to another. As a result, deep analysis and understanding of the EA* and BPMN meta-models (available in annex), represent the main key to start in translation and drawing the links. Table 2 summarizes the mapping of EA* concepts to BPMN concepts. The mapping is accompanied with conditions, which governs the creation of relations between concepts.

Table 2: EA* to BPMN2.0 mapping table (revised version)

EA*	Condition	BPMN2.0
Model		Definitions
Process		Pool, Process, and Participant

EA*	Condition		BPMN2.0	
Extended Activity	Structural		Activity	Sub Process
	Atomic	It is supported by Human		UserTask
		It is supported by IT (no human interaction)		ServiceTask
LogicalOperator	DivergingOr		Gateway	Diverging Exclusive Gateway
	ConvergingOr			Converging Exclusive Gateway
	DivergingAnd			Parallel Gateway
	ConvergingAnd			Parallel gateway
Resource	Material		Data Object	
	Human	Responsible for	Lane	
		Participates in	Resource (added to the list of resources of a task)	
	IT	Responsible for	Lane	
		Participates in	Resource (added to the list of resources of a task)	
Flow	ControlFlow	If the source is an ExternalConnector or InternalConnector and target is an “atomic” ExtendedActivity	MessageFlow	
		If the source is an ExternalConnector or InternalConnector and target is a “structural” ExtendedActivity	Catching Message Event, Message flow, and Sequence Flow	
		If the source is a ProcessConnector or ExtendedActivity	DataObject, and associations	
	OutputInputFlow	If the source is an ExternalConnector or InternalConnector (and target is an atomic Extended Activity)	MessageFlow	
		If the source is an ExternalConnector or InternalConnector (and target is a structural Extended Activity or LogicalOperator)	Catching Message Event, Message Flow, and Sequence Flow	
		If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity or process connector or logical operator)	SequenceFlow	
		If the source is a structural ExtendedActivity or logical operator (and target is an ExternalConnector or InternalConnector)	Throwing Message Event, Message Flow, Sequence Flow	
		If the source is an atomic ExtendedActivity (and	MessageFlow	

EA*	Condition	BPMN2.0
	target is an External or Internal Connector)	
	SupportFlow	If source is a Material resource
		association
Connectors	External	Participant (Pool)
	ProcessConnector	Call Activity
	InternalConnector	Participant(Pool) (Black BOX)

The major changes with respect to the version presented in deliverable D11.3 is reflected in the mapping of “ExtendedActivity” and “Resource” concepts.

Atomic ExtendedActivity

In contrast to “structural” ExtendedActivity, several conditions govern the mapping of “atomic” ExtendedActivity. These conditions vary depending on the type of Resource(s) (if any) supporting the ExtendedActivity:

- **Condition1:** A Human resource is responsible for the realization of the Extended Activity. In this case the atomic Extended Activity is mapped to a UserTask.
- **Condition2:** An IT resource is responsible for the realization of the Extended Activity. In this case the atomic Extended Activity is mapped to a ServiceTask.

Resource

A Material Resource is mapped to a DataObject. The mapping of Human and IT resources depends on the “resourceRole” associated to the SupportFlow connecting it to the ExtendedActivity:

- **Condition1:** the value of the resourceRole is “responsible for”. In this case the resource (Human or IT) is mapped to a lane, in which the supported ExtendedActivity belongs to the lane.
- **Condition2:** the value of the resourceRole is “participates in”. In this case the resource (Human or IT) is added to the list of resources to the supported ExtendedActivity.

ControlFlow

The mapping of a ControlFlow is dependent on the type of source and target connected by the flow:

- **Condition1:** Source is an ExternalConnector or InternalConnector and target is an “atomic” ExtendedActivity. In this case it is mapped to a MessageFlow.
- **Condition2:** Source is an ExternalConnector or InternalConnector and target is a “structural” ExtendedActivity. This case is a “1 to n” relation, in which the “Control” Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.
- **Condition3:** Source is a ProcessConnector or ExtendedActivity. It is a “1 to n relation”, in which the “Control” Flow is mapped to a combination of DataObject and two Associations.

OutputInputFlow

The mapping of an OutputInputFlow is dependent on the type of source and target connected

by the flow.

- **Condition1:** Source is an ExternalConnector or InternalConnector and target is an atomic Extended Activity. In this case the “OutputInput” Flow is mapped to a MessageFlow.
- **Condition2:** Source is an ExternalConnector or InternalConnector and target is a structural Extended Activity or LogicalOperator. This case is a “1 to n” relation, in which the “Control” Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.
- **Condition3:** Source is a ProcessConnector, ExtendedActivity, or LogicalOperator and target is also one of these three options. In this case it is mapped to SequenceFlow.
- **Condition4:** Source is a structural ExtendedActivity or LogicalOperator, and target is an ExternalConnector or InternalConnector. This case is a “1 to n” relation, in which the “Control” Flow is mapped to a combination of MessageFlow, throwing MessageEvent, and a SequenceFlow.
- **Condition5:** Source is an atomic ExtendedActivity and target is an ExternalConnector or InternalConnector. In this case it is mapped to a MessageFlow.

SupportFlow

The mapping of a Flow of type “Support” depends on the type of its source.

- **Condition1:** Source is a Material resource. In this case it is mapped to an Association.

5.3.3 BSM-TIM: Mapping from EA* to UML

Information systems modelling technics argue on the need to describe systems according to different and complementary modelling views. According to general software design and in particular UML, the most important views distinguish **Static**, **Behavioural** and **Functional** view.

At BSM level EA* modelling language is tackling the sequence of actions to be defined and the resources to be involved in the service system. It is clear that it is tackling correctly the functional point of view. From BSM to TIM the mapping described how to obtain the BPMN equivalent information. Nevertheless the process diagram of BPMN is not addressing the data structure of the information to be exchanged between partners. Also it does not take into account the dynamic point of view; the time is not taken into account in the model.

The idea is to identify at BSM in EA* the information that will transformed to UML in order to complete the BPMN at TIM level by UML models that can support a multidimensional view of the system. In conclusion EA* and BPMN are both functional or process oriented, only partial information can be collected from them.

To be clear the main transformation destination language of EA* is BPMN but some additional information that can facilitate the implementation could be also formalised. The first kind of information is about the data structure of the information that circulates during the service creation and delivery; the UML Class Diagram can support this formalisation. The second is the temporal information; the UML State Chart Diagram can support this formalisation. Figure 10 is introducing these complementary models where we distinguish those 2 views.

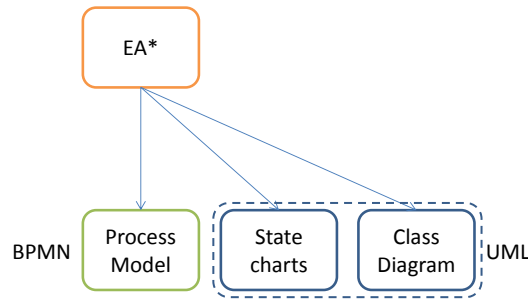


Figure 10: EA* to UML transformations

Matching EA* with UML State Chart Model

We detail in the following table the possible mappings between EA* and UML State Charts. The matching is not systematic, a semantic analysis is required. This analysis can be supported by ontology matching but it is still demanding a human in the loop for validation.

Table 3: EA* to UML2.0 State Charts mapping table

EA*	Condition		UML2.0 State Charts	
Model			Definitions	
Process			Pool, Process, and Participant	
Extended Activity	Structural		Model & State	Coupled Model
	Atomic	It is supported by Human		Atomic Model
		It is supported by IT (no human interaction)		State, Variable states
LogicalOperator	DivergingOr		Temporal constraints can be added to describe the behaviour of the logical operator	State transition rules
	ConvergingOr			
	DivergingAnd			
	ConvergingAnd			
Resource	Material		Behavioural Model including time execution information or time stamped Input Events	
	Human	Responsible for		
		Participates in		
	IT	Responsible for		
Participates in				
Flow	ControlFlow	If the source is an ExternalConnector or InternalConnector and target is an “atomic” ExtendedActivity	Input port / Output Port	
		If the source is an ExternalConnector or InternalConnector and target is a “structural” ExtendedActivity	Input port / Output Port and Input / output Events	
		If the source is a ProcessConnector or ExtendedActivity	Internal Event	

EA*	Condition	UML2.0 State Charts
	OutputInputFlow	If the source is an ExternalConnector or InternalConnector (and target is an atomic Extended Activity)
		Internal Event, External Event
		If the source is an ExternalConnector or InternalConnector (and target is a structural Extended Activity or LogicalOperator)
		Internal Event, External Event
		If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity or process connector or logical operator)
		State transition, internal Event
		If the source is a structural ExtendedActivity or logical operator (and target is an ExternalConnector or InternalConnector)
		State transition, internal Event
		If the source is an atomic ExtendedActivity (and target is an External or Internal Connector)
		State transition, internal Event
	SupportFlow	If source is a Material resource
		Input Port, External Event
Connectors	External	Coupling links
	ProcessConnector	State transition
	InternalConnector	State transition

Matching EA* with UML Class diagram Model

The EA* model is missing the view to represent the data structure of information transported by the process during the creation and delivery of the service. The modeller can refer to OMG website (OMG 2011a) for the UML Class diagram design to support this purpose. It needs to be summarized that the semantic required to build the Class diagram is not already formalized in the EA* model. Some information can come from comments on the EA* model and the others information need to be created from user requirements to be added at TIM level; the reason is more precision is required at TIM level in the model description.

5.3.4 Mapping from TIM level to TSM

The TIM and TSM abstraction levels possess same meta-concepts. This is the reason for the direct mapping (1 to 1 relation) highlighted in Table 4. However, TSM concepts have additional attributes to that of TIM level, therefore models obtained from the transformation process need to be enriched with additional information (possibly with the support of the MSEE reference ontology).

Table 4: TIM-TSM mapping table

TIM	TSM
Service	Service

Process		Process	
Resource	EnterpriseApplication	Resource	EnterpriseApplication
	PhysicalMean		PhysicalMean
	Human		Human
Information		Information	
Organization		Organization	
OrganizationUnit		OrganizationUnit	

5.3.5 TIM-TSM: Mapping from BPMN 2.0 to WS-BPEL 2.0

There are several proposals available for mapping concepts of BPMN2.0 and BPEL, including the one proposed by OMG in the BPMN specification (OMG 2011b – chapter 14).

MSEE is reusing that work, however it is important to understand that when transforming BPMN to BPEL, one is addressing different stages in the life cycle of business process models. Being at the TIM level, the first is used by business analyst/systems architect when designing and improving the business process, whereas technical analysts and programmers use BPEL (at the TSM level) when implementing it. Thus, different requirements exist in different phases.

Moreover, not all BPMN orchestration processes can be mapped to WS-BPEL in a straightforward way. For example, an unstructured loop cannot directly be represented in WS-BPEL. Also, a business process diagram can be made up of a set of (semi-) independent components, which are shown as separate “Pools”, each of which represents an orchestration process. Each of these orchestration processes maps to an individual WS-BPEL process (OMG 2011b – chapter 14).

5.3.6 TIM-TSM: Mapping from BPMN 2.0 to WSDL 2.0

The transformation enabled by the mapping from BPMN to WSDL is complementary to the one of BPMN to BPEL. While BPEL can be used to further model process execution detail at TSM level, WSDL provides a machine-readable description of how the service can be called, what parameters it expects and what data structures it returns.

Moreover, the flow from EA* to BPMN is considered relevant for MSEE, thus the mapping analysis has been focused on the concepts related to the EA* meta-concepts of “Process” and “Activity”. As before, the mapping is not straightforward in all cases and some of the BPMN concepts do not have a direct mapping to WSDL, e.g. “Gateway”. Also, some relations are conditional. For example “MessageFlow” and “SequenceFlow” are only mapped when linked to an “Activity”.

Table 5: BPMN2.0 to WSDL2.0 mapping table

BPMN2.0		Condition	WSDL2.0
Definitions			Description
			Type
			SchemaType
Pool			
Process			Service
			EndPoint
			Binding
Participant			Interface
Activity	Sub Process		Operation
	UserTask		Operation

BPMN2.0		Condition	WSDL2.0
	ServiceTask		Operation
	Task		Operation
Gateway	Diverging Exclusive		
	Converging Exclusive		
	Parallel		
Data Object			
Performer in a “User Task”			
Resource (added to list of task resources)			
MessageFlow		If linked to an Activity.	Input
			Output
			TopLevelElement
Catching Message Event			
DataObject, and association			
SequenceFlow		If linked to an Activity.	Input
			Output
			TopLevelElement
Throwing Message Event			
association			
Call Activity			
Dedicated variable			

5.4 Horizontal Transformations Across the Ecosystem Axis

The work conducted so far under WP11 has been focused more on the service system engineering. Therefore, the vertical transformations along the MDSEA axis of the transformations framework have had the most developments and implementations (see section 7). However, horizontal transformations targeting interoperability and portability are equally important in MSEE. They enable to address cases such as:

- Service system alignment (services matching) between different enterprises;
- Reuse and adapt modelling concepts recognizing possible different semantics in different domains (e.g. Enterprises working in different VRM/Ecosystems);
- Preference of different modelling languages by 2 enterprises (to enable for example cooperation and joint definition of business processes)

Horizontal transformations, assure an effective exchange/sharing of information among different service systems, thus the level of detail of the both source and target models should be similar, and the mapping process more exhaustive. Due to that, greater interoperability benefits but also harder complications are expected in horizontal transformations mapping process. Hence, as addressed before, the use of ontologies might provide an important contribution if the field of horizontal transformations (see Figure 11).

In both transformation types, the level of modelling abstraction of the OMG modelling levels remains unchanged. Both source and target MDSEA models must be an instance of well-defined meta-models, enabling experts to specify mappings that translate any data from one format to the other. Including methods for language translation, context alignment, refactoring of individual models, or even merging different models, this type of transformations occurs along ecosystem integration axis of the transformations framework (Deliverable D11.3). The example illustrated on Figure 11 addresses a case where the mappings are defined based on a pre-existing annotation to domain ontologies, which therefore virtualizes the relation between

both domains. This enables the realization of automatic transformations at BSM level, which will then (4.) enable a different service generation at TIM and TSM levels.

The existence of a high number of parallel research initiatives (see literature review in model transformation, available on deliverable D11.3) provides a solid basis for solutions.

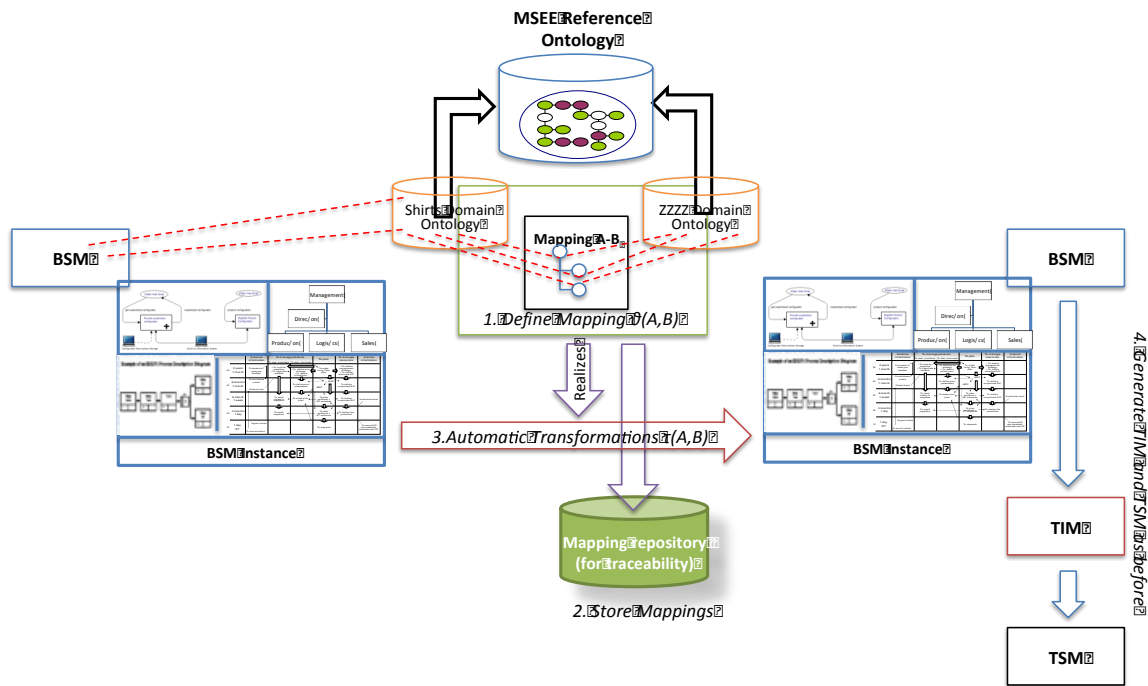


Figure 11: Example of a horizontal transformation process at BSM level

5.4.1 USDL interoperability: Mapping from EA* to USDL

A particular case of horizontal transformations in MSEE can be the integration of the MDSEA SLM models with USDL to, as reported in deliverable D11.5. In the goal to be interoperable with other system from a business and IT point of view, the system should be specified to be compliant with recognized formalisms of the domain. In this context, USDL is used and recognized in the business domain.

The Modelling languages for SLM methods on BSM, TIM, and TSM level differ in their intended usage from USDL Modelling language, i.e., constructs from the MDSEA are used to model the service system with respect to different modelling perspectives during all stages of the SLM service lifecycle, while USDL models are perceived to be mostly relevant in the stage “Service Operations” of the SLM service lifecycle.

The “Service Operations” deals with the phase when a service system is operational for use by customers including service consumption and interaction with customers, monitoring, evaluation, and maintenance.

A detailed mapping table relating EA* with USDL can be found on deliverable D11.5, section 6.3.3.

6 Recommendation for Transformations Implementation in the SLM Toolbox

6.1 Specification for the Transformations Methodology Implementation

6.1.1 Actor Interaction Requirements

Business Expert

Transformations and mappings should be completely transparent to the business expert end-user. He/she only participates to the BSM modelling activities and should be capable to annotate modelling concepts in specific ontologies.

System Architect / System Designer

This type of user should be aware of some transformation details:

- He/she needs to trigger them at the SLM toolbox enabling to navigate from the BSM down-to the TIM level.
- He/she should be able to understand the models at the different levels and annotate modelling concepts in specific ontologies;
- He/she needs to participate in the mapping definition when it is defined explicitly.

Technician/Programmer

Does not require a graphical interface but needs to know all details inherent to the mappings and transformations so that they can be implemented and complemented whenever required.

6.1.2 Functional Specification

The functional specification of the MDSEA transformations in the SLM Toolbox has a direct correspondence to the methodology described in the previous chapter, namely focusing on the part related to the design and execution of the transformation. Nevertheless, according to the WP11 objectives, there are clearly 2 levels of priority:

1. Have the vertical BSM-TIM transformations operational using existing technology and developments published in deliverable D11.3;
2. Complement that work with the use of ontologies and horizontal transformations. Due to the continuity of the work, and possibility for future improvements concerning service systems interoperability as well as dynamic generation of mappings, it is important to enable the connection to ontologies (see section 6.3 “Recommended Plan for Implementation in the SLM Toolbox” for complete detail).

6.1.3 Detailed Technical Specification

Given the context of MDA, QVT is the standard transformation language proposed by OMG. However, considering the languages analysed, ATL has currently the largest user-base and the most extensive information available such as reference guides, tutorials, programmers’ forums, etc. As evidenced by Jouault & Kurtev (2007), it is a largely used language to implement MDA based tools, having a specific development toolkit plug-in available in open source². By all these reasons it was decided to **use ATL to implement model and language transformations** in the scope of the MDSEA transformations framework.

² Eclipse Modelling Project - <http://www.eclipse.org/modeling/>

Although ATL transformation input models can be represented in plain text, it is preferable to **use previously validated XMI³ serialised meta-models conforming to the MOF meta-meta-specification** (level M3 of the OMG modelling levels). However, not all languages selected along deliverables D11.1 and D11.2 have their specifications available as a MOF model (e.g. EA*). They need to be developed.

Concerning implementation, one needs firstly to put the data in an XMI serialisation following the corresponding meta-model specifications. Nevertheless the procedure to do so is not as straightforward as desirable since, even when the inputting model is already XML serialised, it cannot be directly processed by the ATL toolkit, which needs XMI as an input. The complete process is illustrated in Figure 13:

1. **Injection of the original model to an XML MOF meta-model specification** (Figure 12);
2. **Transform that XML format to the source reference meta-model form;**
3. **Actual source-target language mapping.** As explained in the methodology, the relation between the mapping and the ATL can be manual (static implementation) or dynamic, which will require semi-automatic generation of ATL code;
4. **Transform the target meta-model form, back to the XML MOF meta-model specification;**
5. **XML extraction to text.**

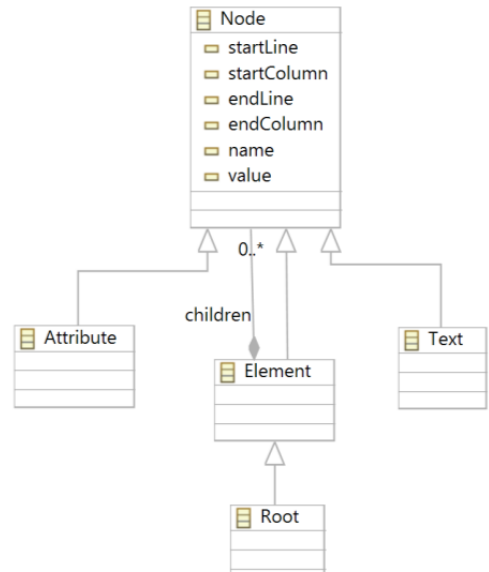


Figure 12: XML Meta-Model (from Rosendal (2005))

To improve the actual source-target mapping, the use of ontologies has been proposed to eliminate semantic problems, and potentiate automation and dynamicity in the mappings definition. Nevertheless, with the exception of the assets ontologies, they don't exist in MSEE with the required maturity. Implementations of the transformations methodology should also **enable the creation of the MSEE reference ontology as detailed in section 5.1.2.**

Finally, and to enable traceability and the semi-automatic generation of ATL code, the transformations methodology envisages the **capability to store explicitly all mappings at a local knowledge base** which, due to the large reasoning potential that can be required in future work, should be implemented **using OWL technology for ontologies**. With a community of thousands of users, Protégé's⁴ internal mechanism for ontologies has been used successfully in many application areas and seems an appropriate choice.

³ XML Metadata Interchange - www.omg.org/spec/XMI/

⁴ <http://protege.stanford.edu/plugins/owl/community.html>

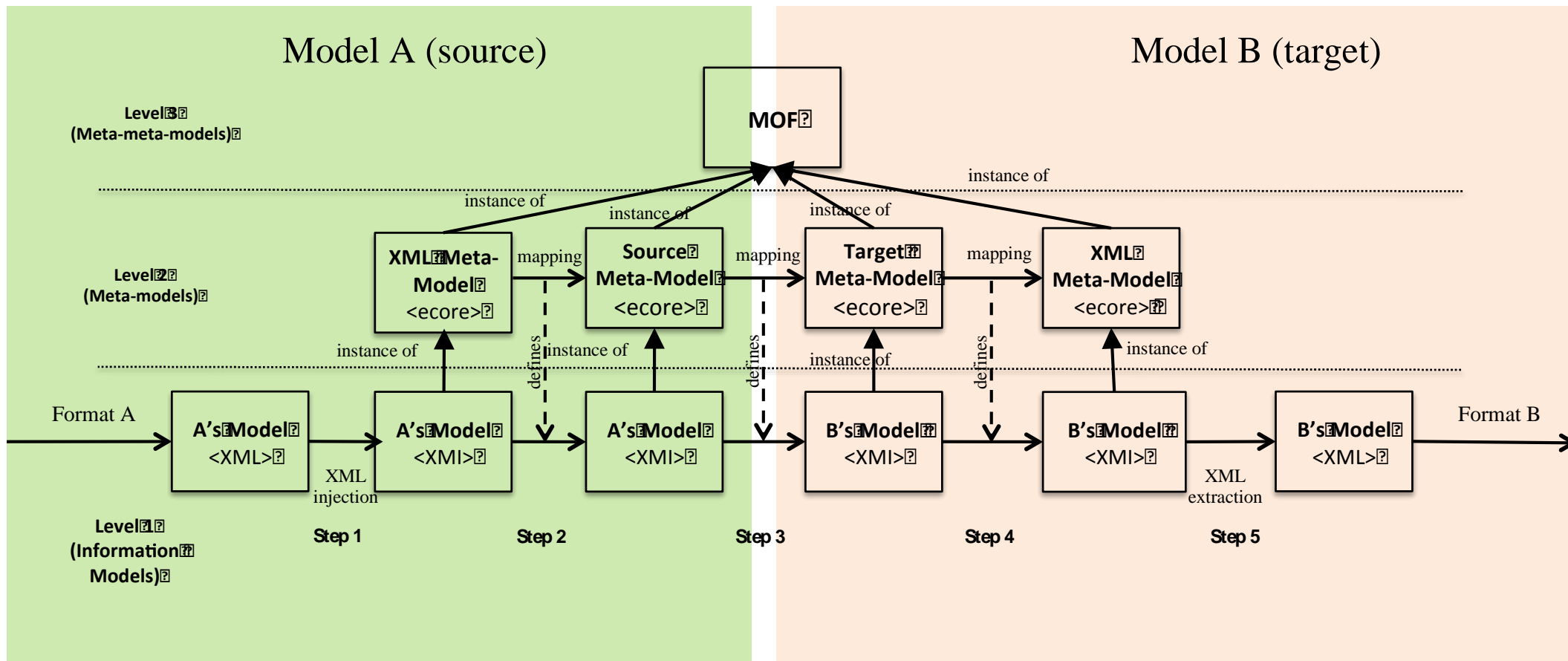


Figure 13: Detailed Transformation

Wrap-up

Summarizing and matching detailed specifications with the different steps of the methodology, the SLM Toolbox should:

- Step 1 of the methodology “Formalize Meta-Models”
 - (Back-end) Use previously validated XMI serialised meta-models conforming to the MOF meta-meta-specification;
- Step 2 of the methodology “Build the MSEE Reference Ontology”
 - (Back-end) Generate MDSE core concepts in OWL through an automatic transformation
 - (Front-end) Enable the definition of links/annotations to external knowledge sources through a remote connection
 - (Front-end) Enable to select models (or parts of) to contribute to a domain ontologies (OWL transformation)
- Step 3 of the methodology “Define the Mappings”
 - (Front-end) Visualize and build mapping tables (suggested externally through intelligent agents or manually defined)
 - (Back-end/Front-end) Enable remote connection to knowledge sources (MSEE Reference Ontology)
 - (Back-end) Capability to store explicitly all mappings at a local knowledge base using OWL technology for ontologies
- Step 4 of the methodology “Implement the Mappings”
 - (Back-end) Use ATL to implement model and language transformations
 - (Back-end) Inject the original model to an XML MOF meta-model specification
 - (Back-end) Map that XML format to the source reference meta-model form;
 - (Back-end) The relation between the mapping and the ATL can be manual (static implementation) or dynamic, which will require semi-automatic generation of ATL code;
 - (Back-end) Map the target meta-model form, back to the XML MOF meta-model specification;
 - (Back-end) XML extraction to text.
- Step 5 of the methodology “Execute the transformation”

Implementation of these functionalities is directly associated with the plan of section 6.3.

6.2 Horizontal Transformations Across the Ecosystem: How to include in the SLM Toolbox

The SLM Toolbox is a modelling tool that supports the design of the service system. In this context, horizontal transformations and ontologies are traditionally mostly important to enable activities such as:

- Collaborative design (e.g. processes), thus enabling to import and integrate/interoperate models initially specified using different languages or paradigms;
- Translation of modeling concepts to different domains, thus enabling a better analysis of the models by domain specialists. Could be important in the case that a specific

enterprise operates at different ecosystems and needs to reuse part of the service system models;

In MSEE, another case that can be considered as a requirement for horizontal transformations is the integration of BSM/TIM/TSM models with USDL to enable service matching. The main purpose of USDL models is to model a set of selected service aspects that are relevant for activities like service description, service discovery, service composition or aggregation, and service consumption / service participant interaction. As described in deliverable D11.5, the modelling tool should be able to send / inject some information (for example, partner name, address, service functions etc.) into USDL database.

6.3 Recommended Plan for Implementation in the SLM Toolbox

In order to provide a specific recommendation to the SLM Toolbox, the tables bellow provide an indication on which functionalities can be integrated and when. They clearly distinguish what was done at the time of D11.3 (M8), what is being done at the time of D11.4 (M15), at the time of the review (M18), and what can be done until the conclusion of the toolbox (M24), and towards the future (outside MSEE).

It is important to note that a specific SLM Toolbox workshop to discuss M24 implementation requirements is going to be held in Bordeaux from the 28th of February to the 1st of March 2013, thus this plan can be changed.

Cells marked in Grey are not envisaged to be part of the SLM Toolbox functionalities.

Table 6: Plan for modelling/ontology support functionalities

Modelling \ Functionality		Meta-Model Formalization	MDSEA Modelling	Ontology Annotation
BSM	Core Concepts	-	M15	M24
	EA*		M8	M24
	Other			
TIM	Core Concepts		M15	M24
	BPMN2.0		M15	M24
	Other		UML (Future)	UML (Future)
TSM	Core Concepts		-	-
	WS/BPEL			
	WSDL2.0			
	Other			

Table 7: Recommended plan for transformation functionalities

Functionality Transformation			Ontology Building	Support to Mapping Visualization/ Definition	Explicit Mapping Storage	Ontology Annotation (Mapping)	Static Transformation Implementation	Dynamic Transformation Implementation	Execute the Transformation
Vertical	BSM-TIM	Core Concepts	-	Future	M24	Future	M18	Future	M18
		EA* - BPMN2.0		Future	M24	Future	M15	Future	M15
		EA* - UML		Future	M24	Future	M24	Future	M24
		Other							
	TIM- TSM	Core Concepts	-						
		BPMN2.0 – WS/BPEL							
		BPMN2.0 – WSDL2.0							
		Other							
Horizontal	BSM Integration with USDL		-	Future	M24	Future	M24	Future	M24
	TIM Integration with USDL			Future	M24	Future	M24	Future	M24
	TSM Integration with USDL		-						
	BSM Integration with OWL (ontology)		M24	Future	M24	Future	M24	Future	M24
	TIM Integration with OWL (ontology)		M24	Future	M24	Future	M24	Future	M24
	TSM Integration with OWL (ontology)		-						
	Other								

7 Specific Implementations

The transformations framework that has been incorporated into MDSEA (summarized in section 4.3) can be applied to perform both vertical and horizontal transformations, using specific MDA/MDI tools to translate from one abstraction level to another.

Following the architecture of Figure 14 (instantiation of the generic one proposed in deliverable D11.3) and applying the mappings presented in section 5.3 (result of the third step of the transformations methodology), some ATL transformations have been implemented and executed (steps 4 and 5) in the eclipse environment⁵. They demonstrate the validity of the methodology, as well as the transformations framework and architecture proposed in deliverable D11.3.

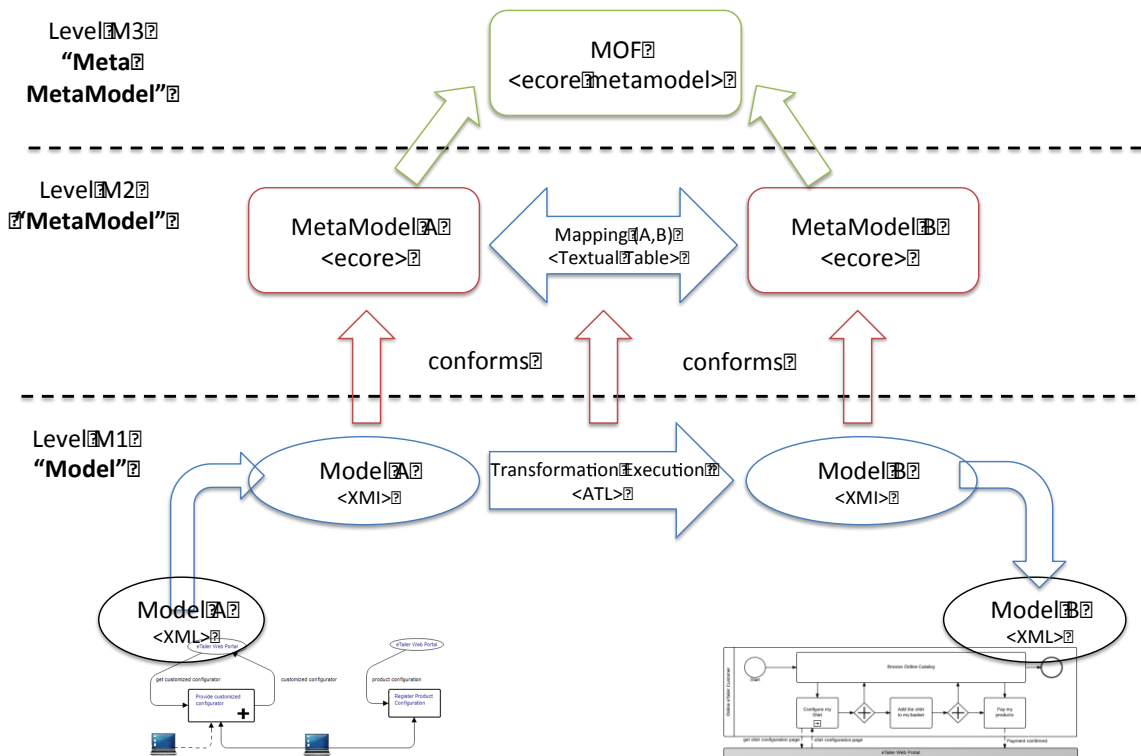


Figure 14: Transformations architecture used in implementations

Starting from an XML serialization of the model and compliant ecore representation of its meta-model, it is possible to write and execute ATL code following specific mapping tables. This architecture follows the detailed specifications for transformations for the SLM Toolbox (section 6.1.3), thus it is easily integrated in that environment whenever required.

Continuing a similar approach as in deliverable D11.3, advances on the BSM to TIM and TIM to TSM model transformations are considered to be an important milestone for this deliverable. Thus, this chapter includes specific implementation examples of vertical transformations to demonstrate the full cycle of transformations from BSM to TSM. They are, at this time the most relevant for MSEE, enabling to model the different service systems for each Use Case.

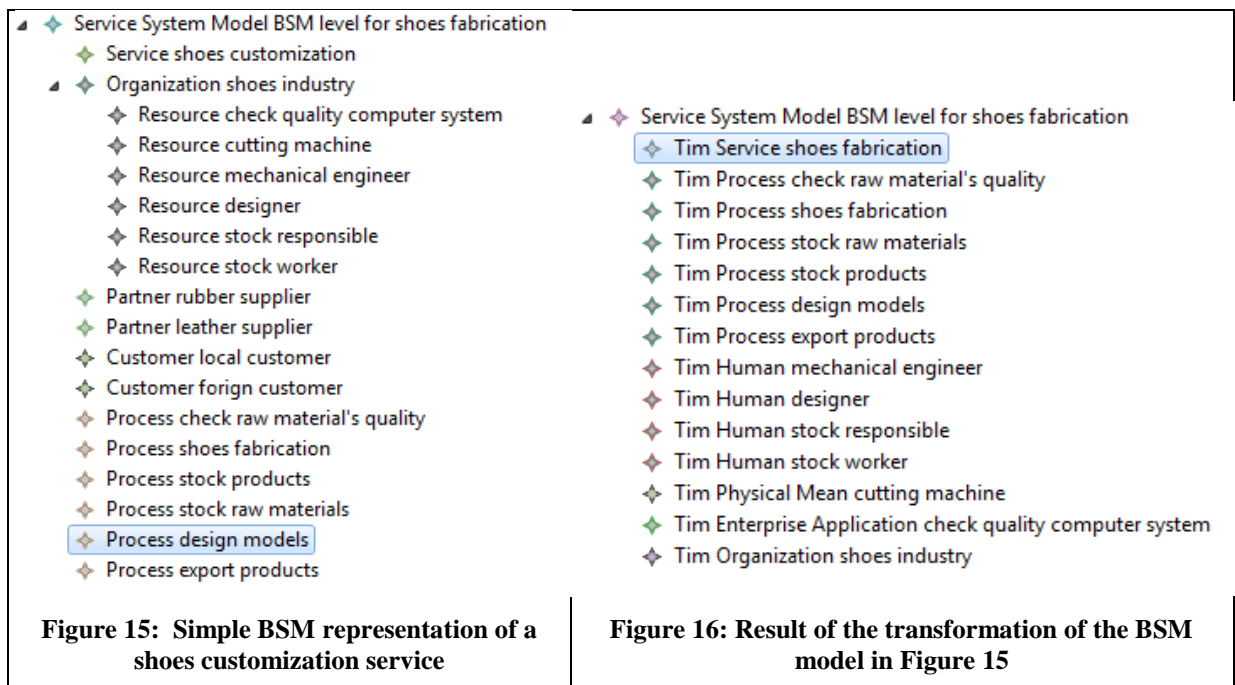
⁵ <http://www.eclipse.org>;
<http://www.eclipse.org/atl/>

7.1 Vertical Transformations (BSM to TIM)

7.1.1 Core Concepts

The two figures below were developed and viewed using the SLMTToolBox. The first, Figure 15 is a simple BSM representation of a shoes customization service, where we can view partners involved, organization and its resources, processes needed, and the service's customers.

The second, Figure 16 is the result of the transformation of the BSM model into TIM. By examining the two figures it is possible to notice that information concerning customers is not revealed at TIM Level, as expected by the mapping proposed on section 5.3.1.



7.1.2 Transformation from EA* to BPMN 2.0 (Vertical – language level)

Full details related with the implementation of EA* to BPMN 2.0 can be found in deliverable D11.3. Since then, the transformation has been perfected but the implementation details are similar and will not be here repeated.

7.2 Vertical Transformations (TIM to TSM)

7.2.1 Transformation from BPMN to WS-BPEL (Vertical – language level)

Following the existing mapping between BPMN2.0 and BPEL (OMG 2011b – chapter 14), several implementations are also available, namely:

- Transformation for Eclipse BPEL plugin (<http://code.google.com/p/bpmn2bpel/>);
- Oracle JDeveloper and Business Process Analysis Suite (refer to <http://www.oracle.com/technetwork/articles/dikmans-bpm-101437.html>).

MSEE is analysing that work for integration in the SLM Toolbox. Concerning the transformation itself, despite being quite similar, the differences in the mapping leads to more

complex BPEL results. Particularly, the more flows and loops in a BPMN diagram, the more differences there will be in the corresponding BPEL.

7.2.2 Transformation from BPMN to WSDL (Vertical – language level)

Focusing on the mapping of “Participant” - “Interface”, and “Activity/User Task” – “Operation” concepts (presented in section 5.3.6), the following snapshots (Figure 17) illustrate the transformation execution. The example used is not specifically related to any MSEE use case.

Starting from the top (BMPN XML representation), “Participant_2” with the name “Non-initiating Pool” is linked with “Process_1”, which is graphically represented by “Lane Set 1” (on the left). It is possible to see, among other things that “Process_1” contains a number of activities, in particular a User Task named “edit 1st level ticket”.

Then, applying the rules “Interface” and “Operation Activity” (both rules are linked and the first calls the second) to the BPMN model, it is possible to obtain the desired output WSDL model (bottom left of the figure). In this case one see that:

- Due to the first rule, the “Participant” has been transformed into and “Interface” element;
- Due to the second rule, the “Activity”, instantiated as a “User Task” has been transformed in an “Operation”.

Full details of the transformation rules are not included in the document to maintain readability and minimize space occupation.

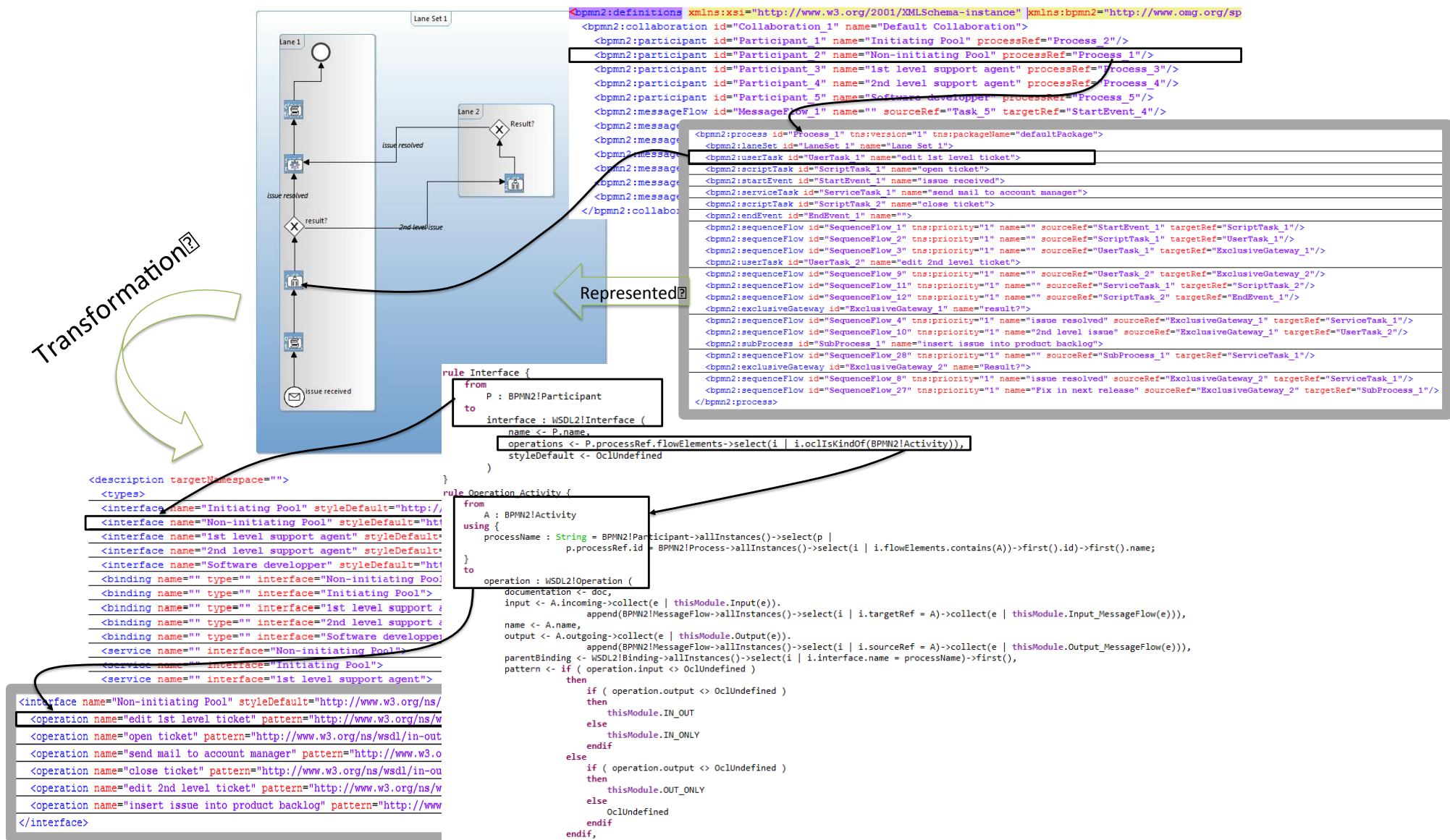


Figure 17: BPMN 2.0 to WSDL 2.0 Transformation Example

8 Integrated SLM Toolbox Transformation Example from the Bivolino Use-Case

The Figure 18, below, illustrates a scenario from the Bivolino Use-Case indicating the steps to be taken in order to transform the Bivolino's customized configurator EA* model into a collaboration BPMN 2.0 model. Deliverable D15.3 "Specifications and Design of SLM Platform" also is using the same example, thus more details apart from the transformation can be found there.

The EA* diagram represents the user system interaction, and due to transformation requirements this diagram should be manually decomposed (steps 1 and 2) into two separate diagrams: user centred and system centred. After this decomposition, automatic model transformation is applied to both diagrams (steps 3 and 4). The result of these steps will be two separate BPMN diagrams, which are user and system centred. Then the BPMN diagrams are joined manually, in order to represent the user system interaction in one collaboration diagram (step 5). Later this diagram is enriched manually at the TIM level for a more detailed model.

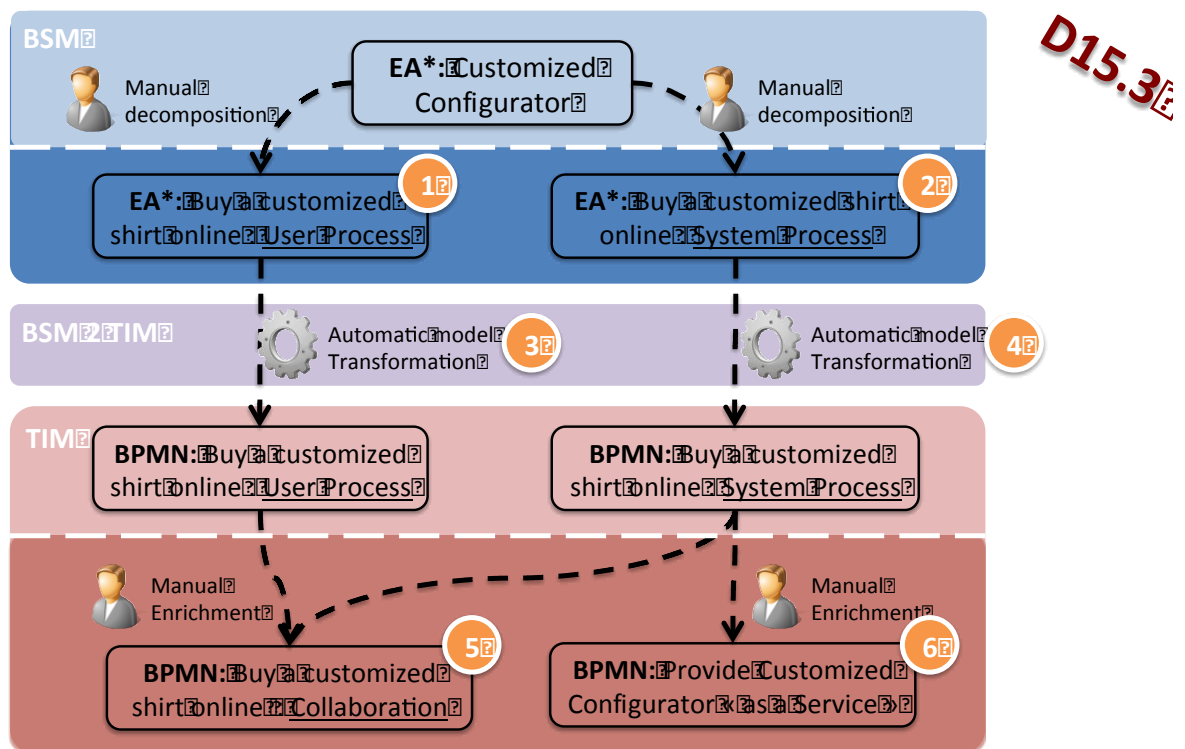


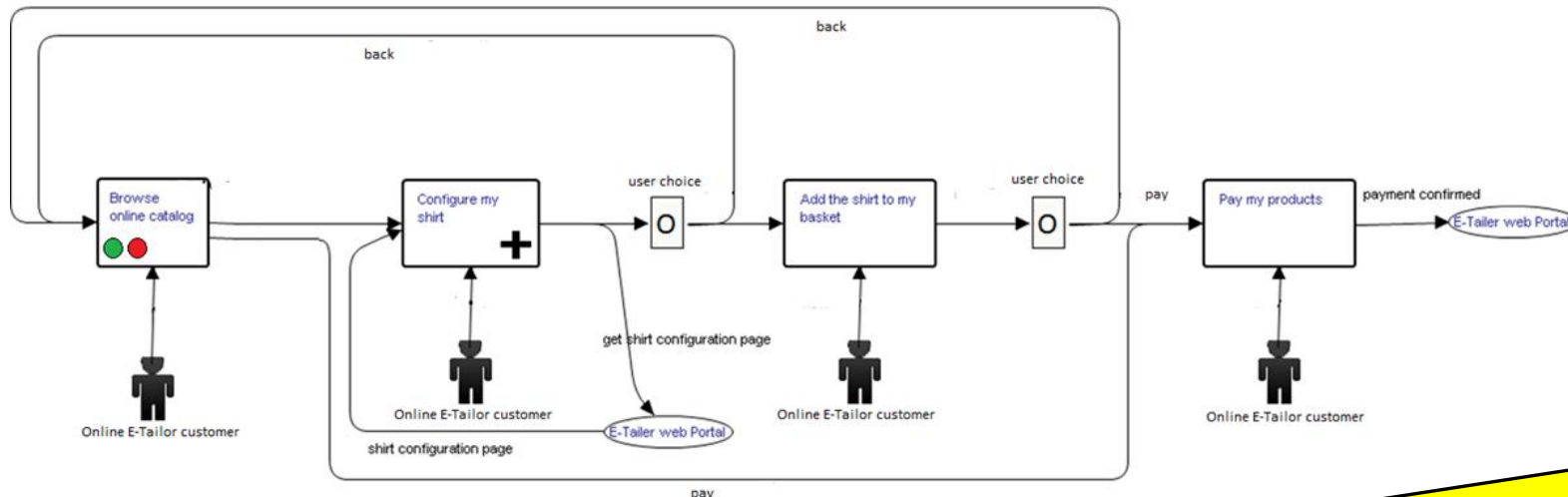
Figure 18: Bivolino modeling and transformation scenario (see future deliverable D15.3 for more detail)

Snapshots are taken at each step (see sequence below), thus helping to illustrate the scenario and the modelling activities conducted after the transformation. To note that this example is complying with the first priority specifications, i.e. to have the vertical BSM-TIM transformations operational using existing technology and developments published in deliverable D11.3, and no work with ontologies has yet been implemented in the SLM Toolbox.



«Buy a customized shirt online» User Process

Goal to describe the activities performed by an online customer when buying a custom shirt, on the E-Tailer web portal

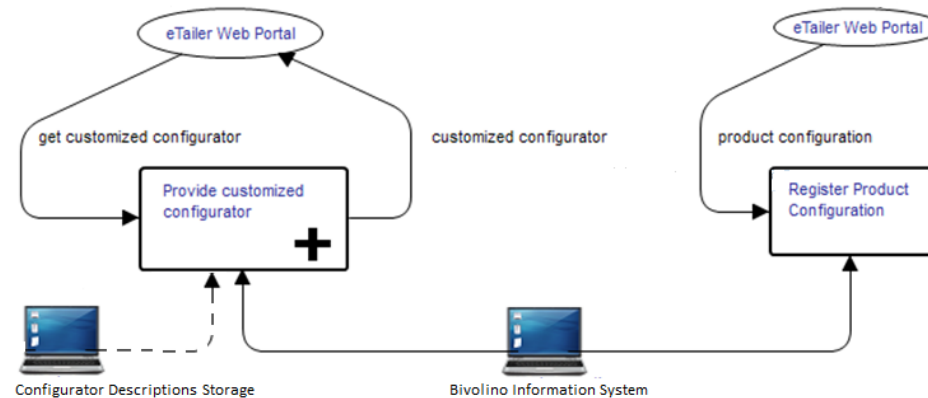


Extended
Actigram Star

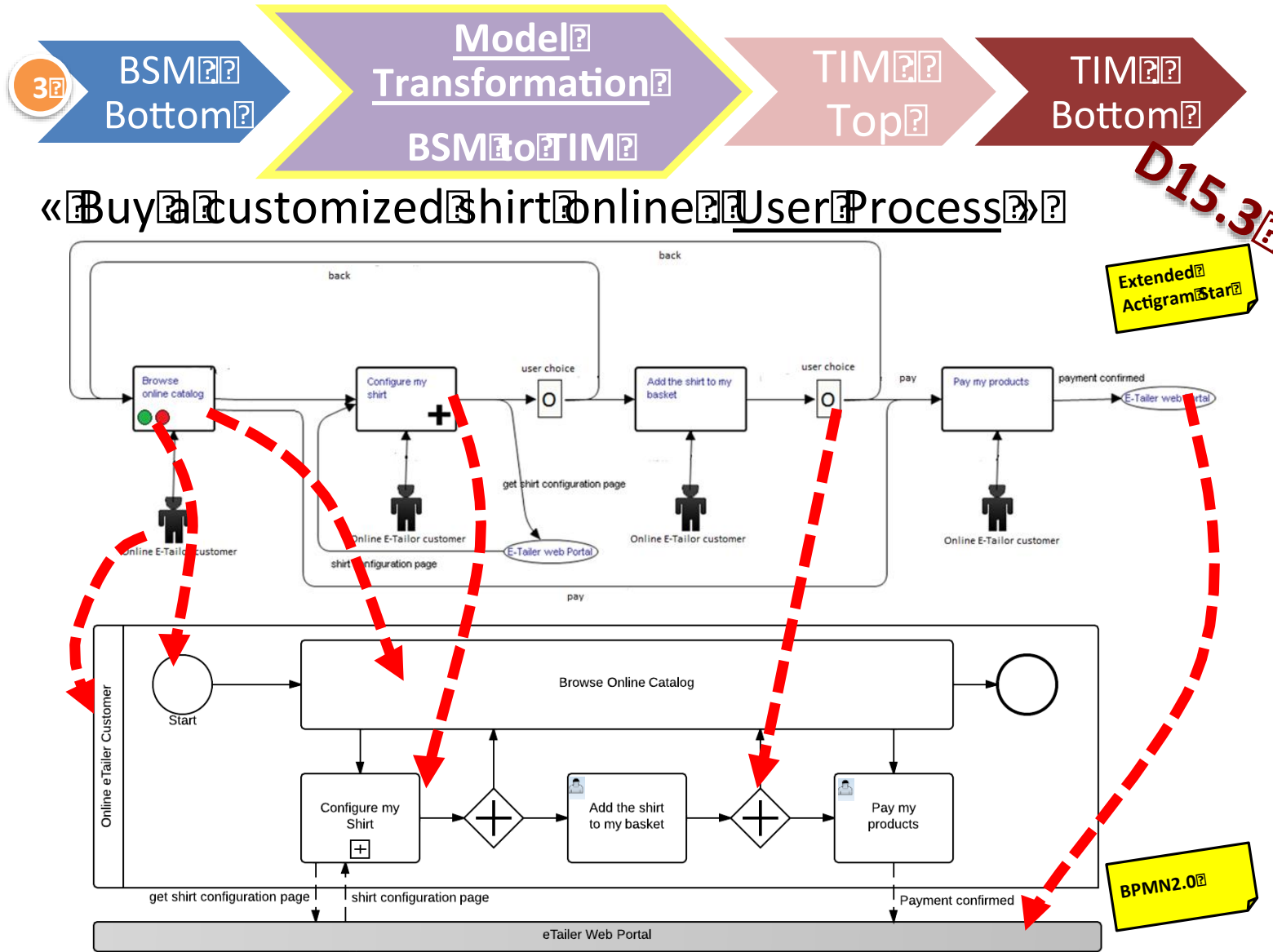


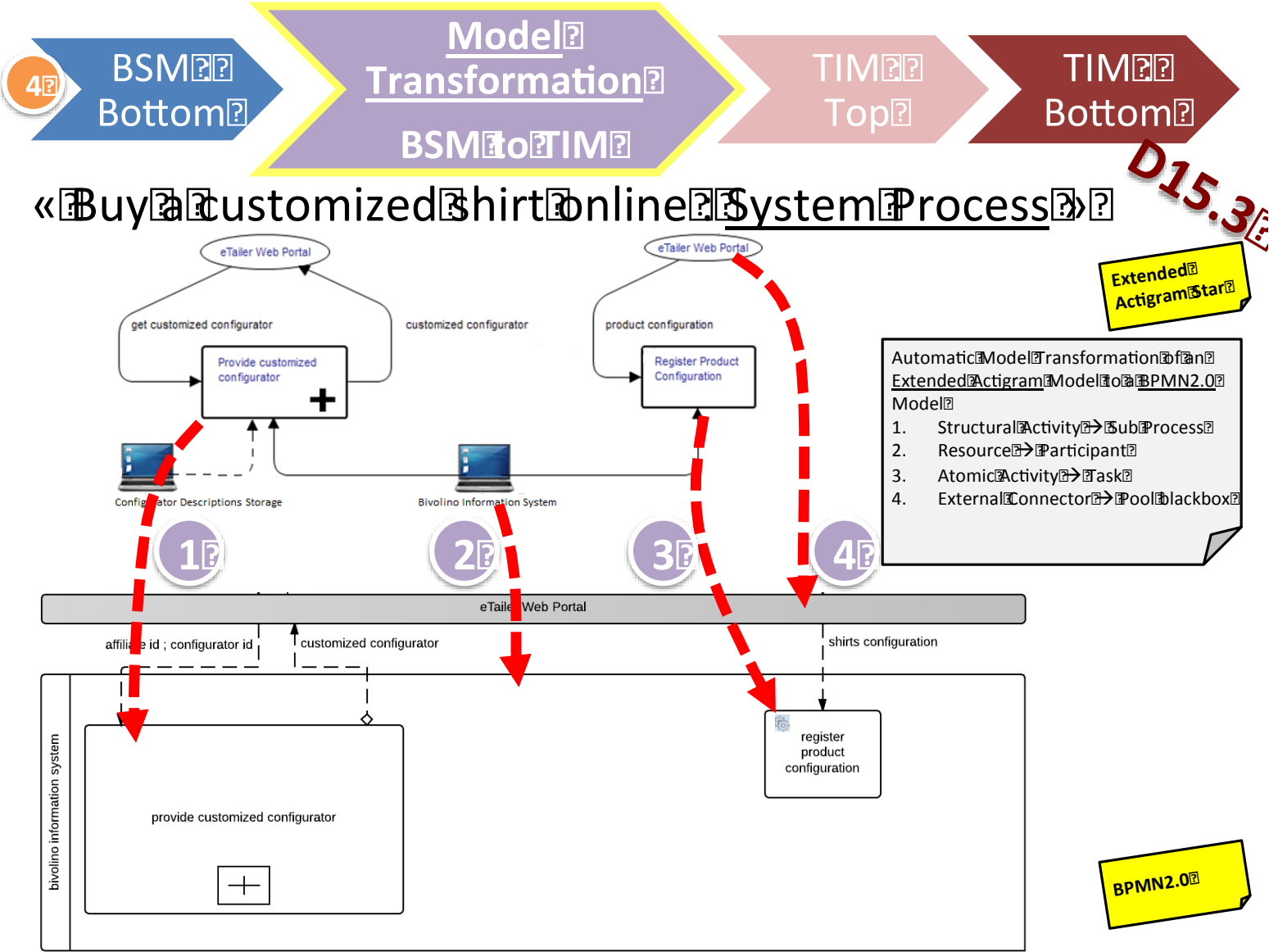
«Buy a customized shirt online» System Process

Goal to describe the activities performed by the system, when an customer buys a custom shirt online



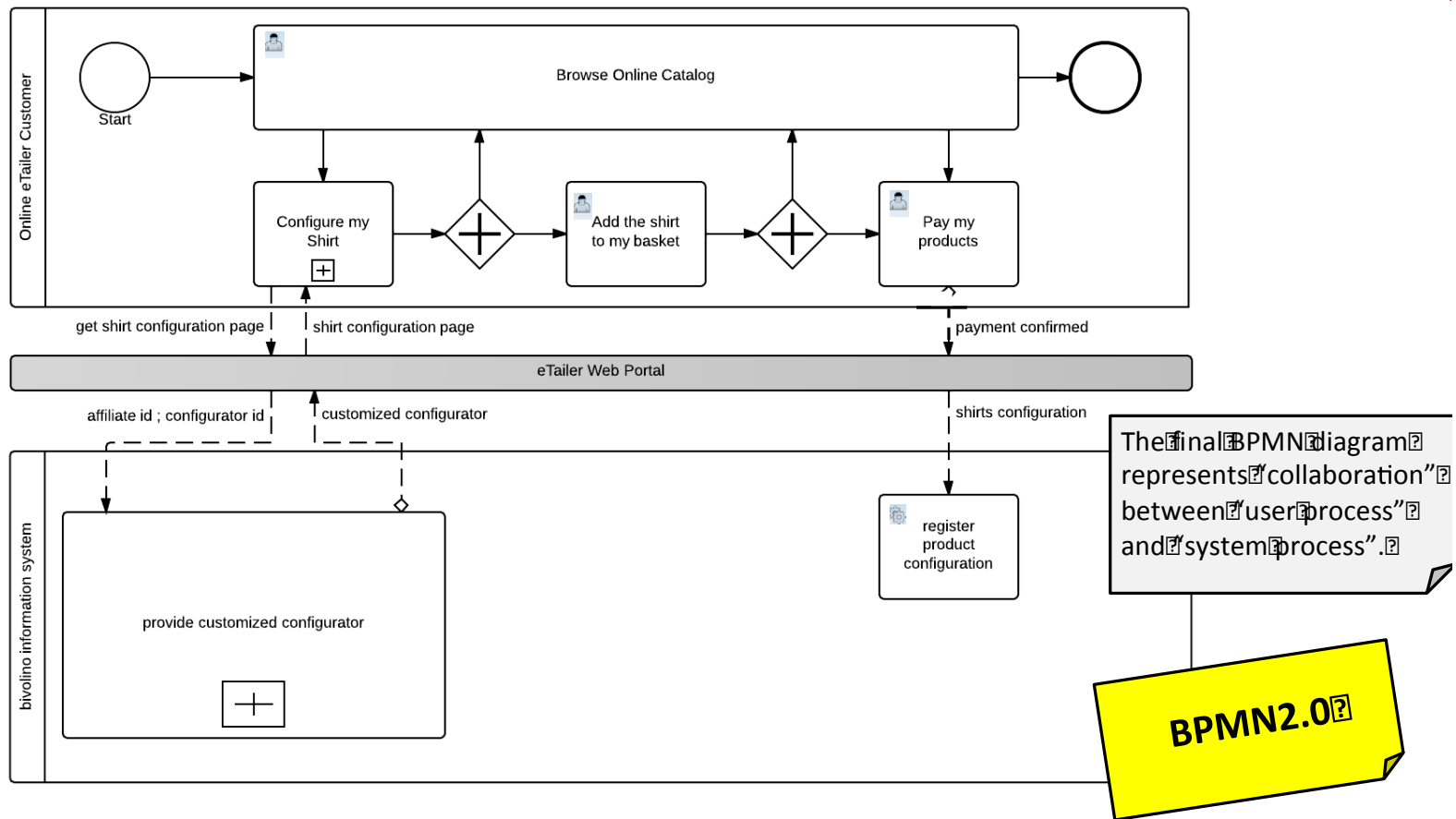
**Extended
Actigram**







«Buy a customized shirt online» Collaboration





«BPMN: Provide Customized Configurator as a Service»

D15.3

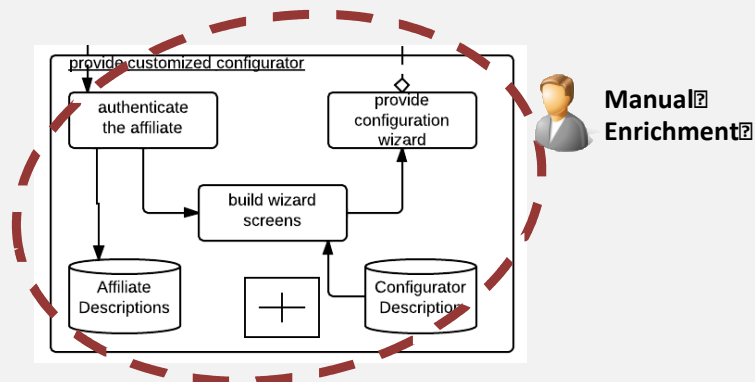


Figure 19: Sequence of snapshots illustrating the Bivolino modeling and transformation scenario (see future deliverable D15.3 for more detail)

9 Conclusions

This deliverable has permitted to explicit aspects of the MDA/MDI transformation framework that were not tackled in previous WP11 documents. The proposed transformations methodology is flexible providing specifications along 2 levels of priority: 1) To have the vertical BSM-TIM transformations operational using existing technology and developments; 2) Complement that work with new research using ontologies and horizontal transformations techniques.

Based on model transformations, the MDSEA has the challenge to unify every step of the enterprise (or virtual enterprise) servitization from its start as a BSM of the services' business requirements reflecting the company strategy, through TIM defined functions and behaviour from the perspective of IT, Human and physical means components, to one or more TSMs, down to generated code and deployable services. Also, MDSEA answers to the challenge of portability and interoperability, enabling enterprises from complementary domains and using different technologies to collaborate in the frame of the ecosystem.

The MDA/MDI transformation methodology integrates previous research developments and provides concrete specifications on the steps to be taken from the transformation preparation until its execution. Ontologies are considered relevant but not mandatory for the success of transformations. They are analysed in this deliverable as parseable knowledge repositories that can support automation, by enabling intelligent services to work on top of them and providing the Human actor additional semantics throughout a certain activity. As explained, intelligent agents reasoning on this knowledge can even suggest an initial set of dynamic mappings.

Following the architecture and applying the mappings presented, some ATL transformations have been implemented and executed. They demonstrate the validity of the methodology and are supported by a concrete example from the Bivolino use-case, highlighting how the transformations are needed on a real scenario.

Concerning future work, the MDA/MDI model transformations activity will support implementations along the project, namely the SLM Toolbox, as indicated in the Recommended Plan for Implementation in the SLM Toolbox . It is important to note that some functionalities are considered not fundamental for the MSEE progress, thus are marked as future activities (might occur after the project – exploitation stage), and a specific SLM Toolbox workshop to discuss M24 implementation requirements is going to be held in Bordeaux from the 28th of February to the 1st of March 2013, which might revise that plan.

10 References

- Agostinho, C., Sarraipa, J., Gonçalves, D., & Jardim-Goncalves, R. (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. DOCEIS'11. Costa de Caparica
- Athena IP (2006) ATHENA Interoperability Framework (AIF). Available at: <http://modelbased.net/aif/index.html> [Accessed January 20, 2012].
- Czarnecki, K. & Helsen, S. (2006) Feature-based survey of model transformation approaches. IBM Systems Journal, 45 (3), p.pp.621-645. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5386627>
- Doumeings G., Vallespir B., Chen D. 1998: GRAI grid, decisional modeling ; in Handbook on Architecture of Information System ; edited by P. BERNUS, K. Mertins, G. SCHMITH ; International Handbook on Information Systems ; Springer Verlag
- ebPML (2007) WS-BPEL 2.0 Metamodel. Available at: <http://www.ebpml.org/wsper/wsper/ws-bpel20b.png>
- Jouault, F. & Kurtev, I. (2007) On the interoperability of model-to-model transformation languages. *Science of Computer Programming*, 68, pp.114–137
- OMG (2003) *MDA Guide Version 1.0.1 (omg/2003-06-01)*, Object Management Group. Available at: <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf> [Accessed September 7, 2011].
- OMG (2011a) OMG Unified Modeling Language™ (OMG UML), Infrastructure - version 2.4.1. Available at: <http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/>.
- OMG (2011b) Business Process Model and Notation (BPMN), version 2.0 - formal/2011-01-03. Available at: <http://www.omg.org/spec/BPMN/2.0>
- Rosendal, P. (2005) XML 1.1 [Internet]. Available from: <http://www.emn.fr/z-info/atlanmod/index.php/Atlantic>
- Wang, Wenguang, Tolk, A. & Wang, Weiping (2009) The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S. In *Spring Simulation Multiconference (SpringSim'09)*. San Diego, CA, USA.

Annex A – MDSEA Core Concept Meta-Models

BSM Meta-Model

Product, Service, Functionality, Resource, Organization, Decision, and Process are among the major concepts and core constructs used at the Business Service Modelling level. Based on the BSM templates presented in deliverables D11.1 and D11.2, the following Figure 20 represents the BSM core constructs meta-model. It is a package of abstract classes that can be extended by specific 1:1 realization relationships with the language specific constructs.

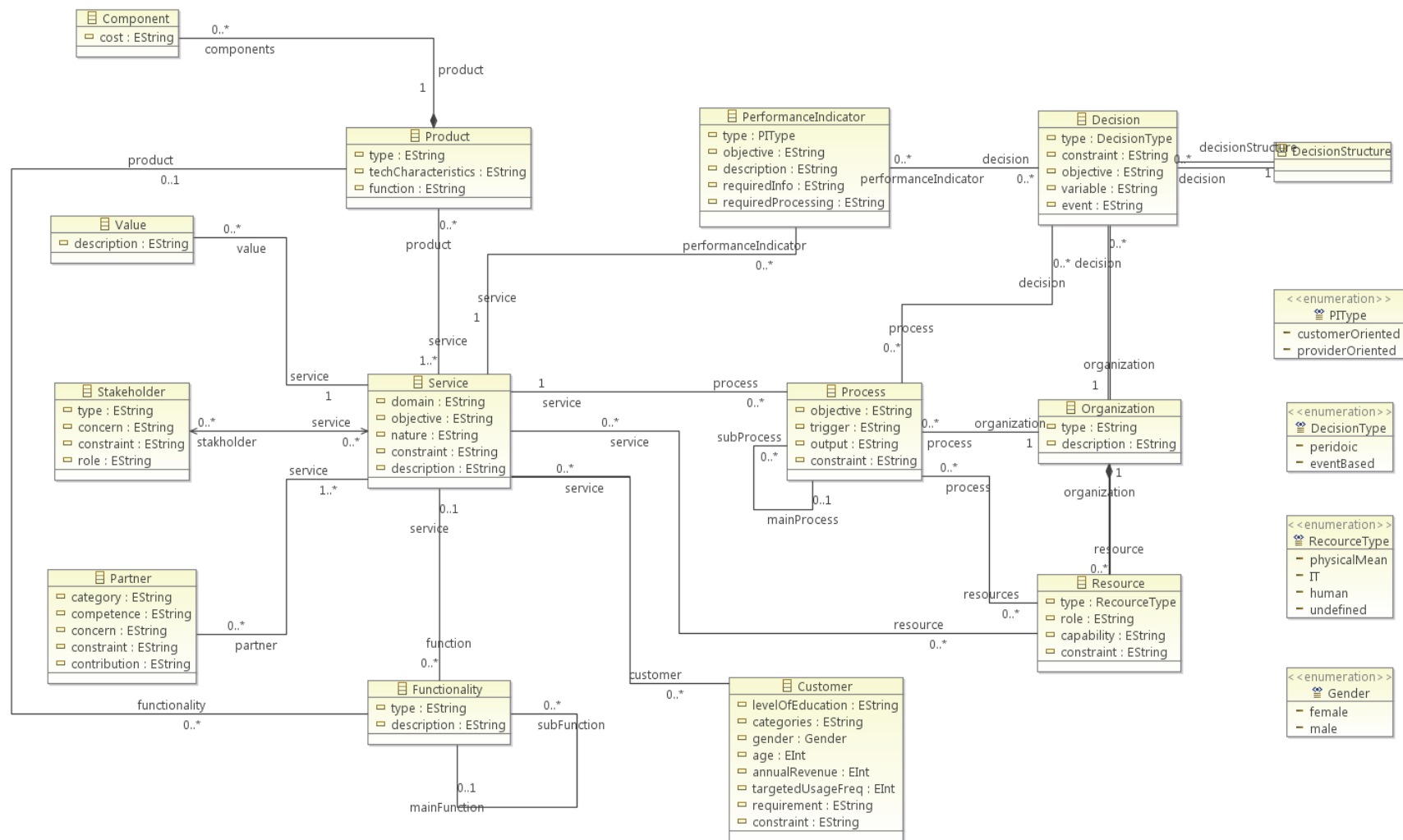


Figure 20: BSM Core Concepts Meta-Model

TIM Meta-Model

Process, Service and Resource form the common concepts package used at the Technology Independent Modelling level. The modeling at TIM level aims at specifying three types of resources (IT, Organization/Human and Physical mean) that constitute a Service System.

As before (BSM), based on the TIM templates presented in deliverables D11.1 and D11.2, the following Figure 21 represents the TSM core constructs meta-model. It is a package of abstract classes that can be extended by specific 1:1 realization relationships with the language specific constructs.

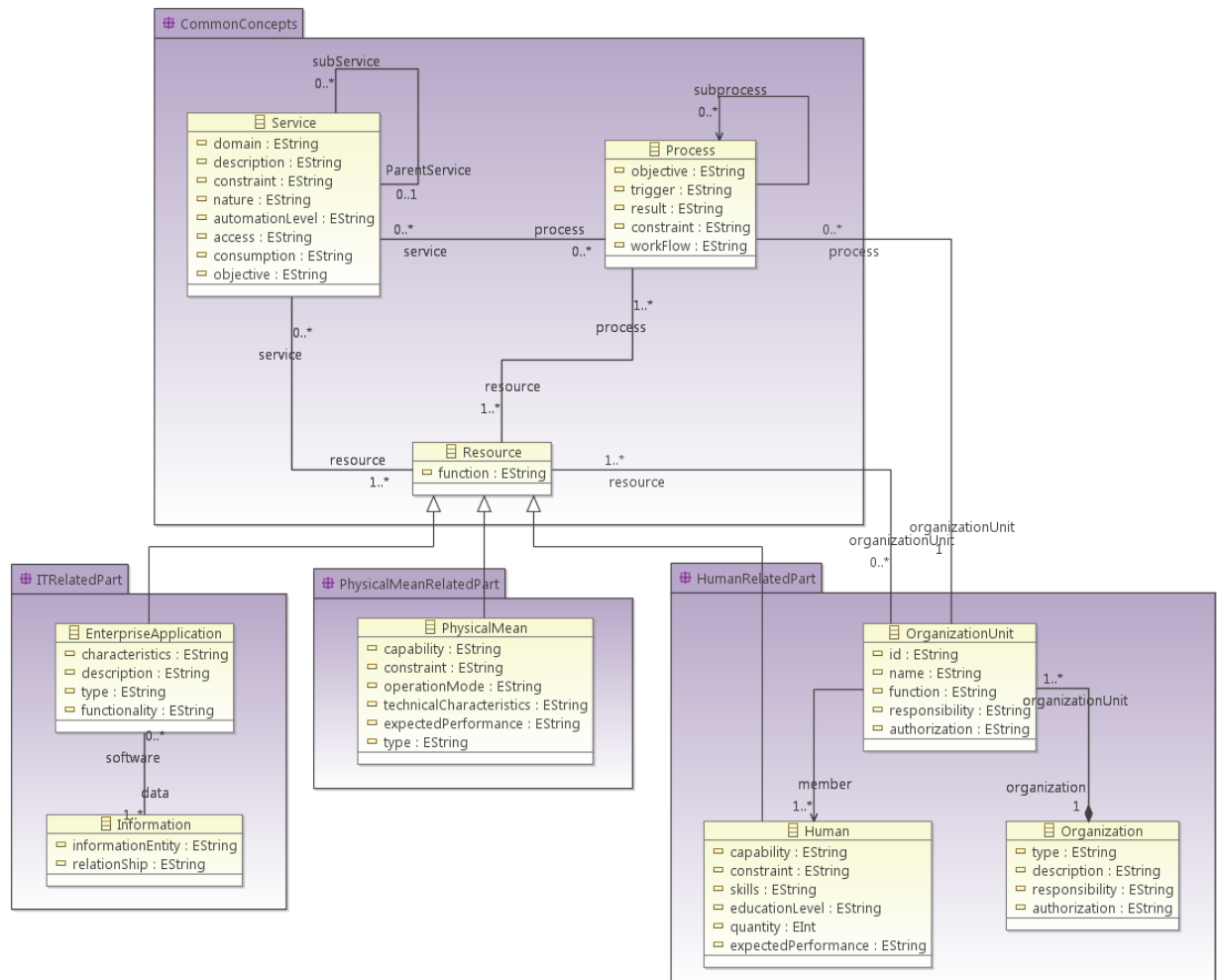


Figure 21: TIM Core Concepts Meta-Model

TSM Meta-Model

The TSM meta-model presented in Figure 22 is a meta-model conceptually very similar to the TIM, extended with additional information specific to a particular implementation. Most of TIM constructs are refined and detailed at TSM level with additional attributes or formalisms (refer to deliverables D11.1 and D11.2 for complete details).

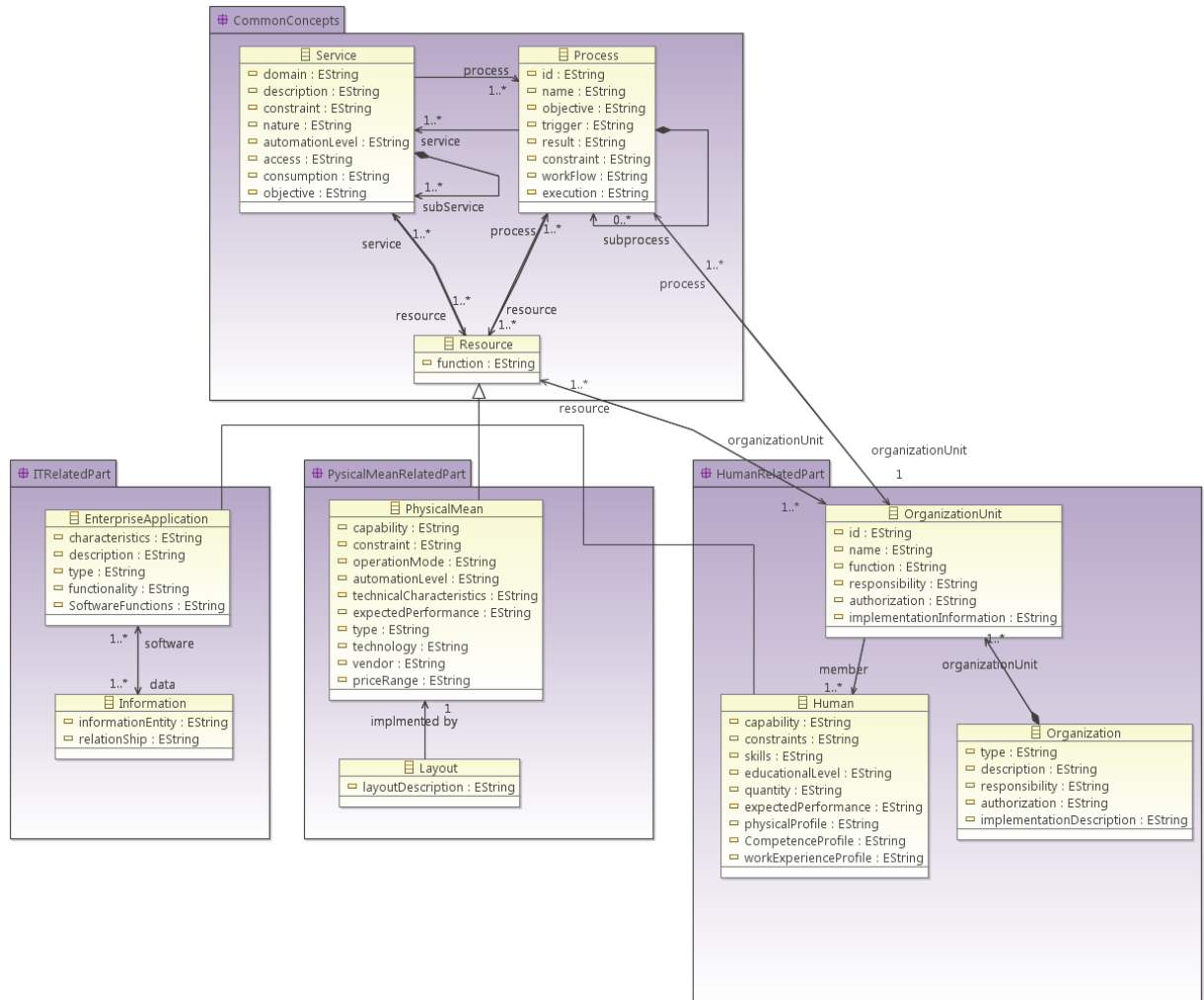


Figure 22: TSM Core Concepts Meta-Model

Annex B – Selected Languages Meta-Models

Extended Actigram Star (EA*)

Extended Actigram Star (EA*) relies on previous work developed in the frame of the GRAI Methodology (Doumeingts G., Vallespir B., Chen (1998)), which defines “GRAI Extended Actigram” as a process modelling language, among other graphical formalism, for enterprise modelling and “decision centric” analysis. The goal of Extended Actigram Star is to:

- Provide a common and simple modelling notation that is understandable by business users, for the description of business process.
- Reduce the gap between the ideation and the design of business process (by its simple and accessible syntax).
- Facilitates the transformation of business process models toward other structured modelling languages offering more detailed constructs (for instance BPMN2.0).

EA* conceptual meta-model is formed of several regrouping levels which permit to generalize concepts and reduce and factor out details. Thus, EA* elements are divided into three sub packages:

- **Root package:** contains the root element of the EA* Language (Model)
- **General Elements package:** this package tries to reduce and factor out details so that the focus is on the general elements that form this language. General concepts package’s goal is to generalize without entering into details.
- **Core Elements package:** this package contains the elements which form an Extended Actigram Star diagram, where every concrete element has a corresponding graphical representation defined by this language.

All Extended Actigram Star elements inherit from the BaseElement class three common attributes: **id**, **name**, and **code**.

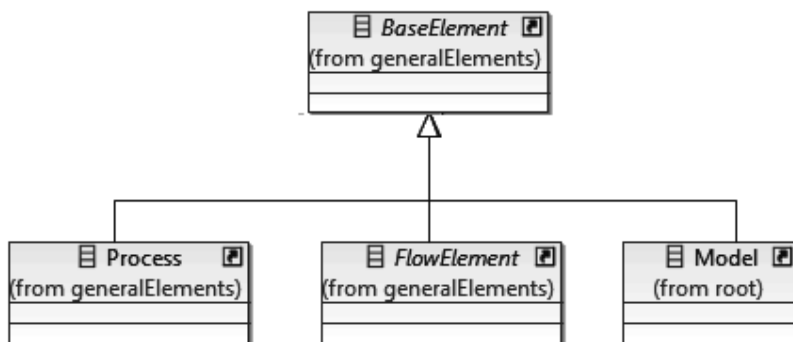


Figure 23: BaseElement

Extended Actigram Star diagram is a representation of a business process (the subject to be modelled). A **Process** is composed of FlowElement(s), which is an abstract representation of all elements constituting the diagram.

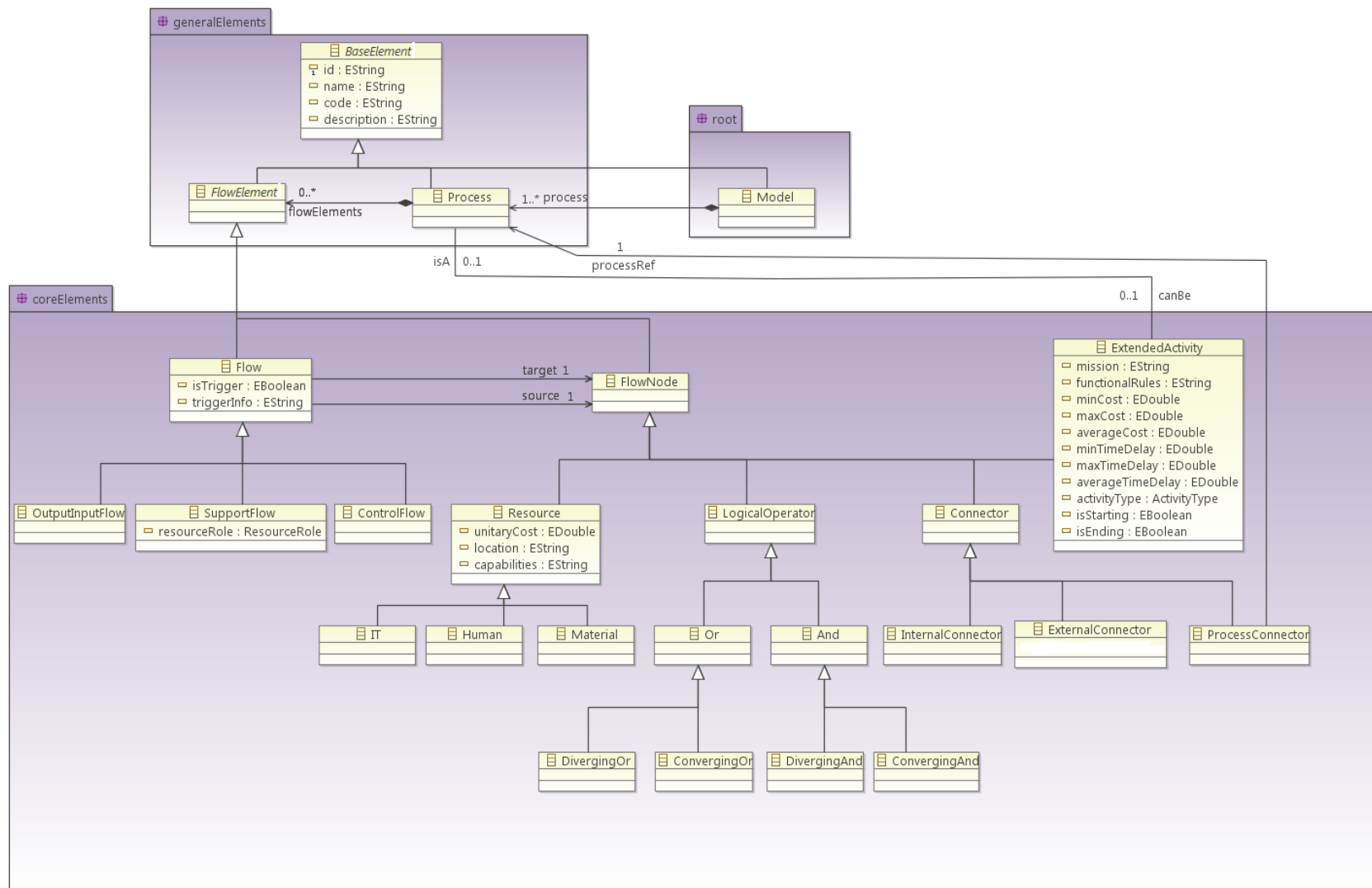


Figure 24: EA* Conceptual Meta-Model

A **FlowElement** can be a:

- **Flow:** used to link FlowNodes. A Flow is directed and can be an OutputInputFlow, Control Flow or SupportFlow. Several constraints governs the connection of a flow, depending on its nature and type of its source and target nodes. A Flow has one source and one target.
 - OutputInputFlow: depicts the logical sequence between two elements.
 - ControlFlow: indicates that the flow is carrying the control of the realization of an ExtendedActivity
 - SupportFlow: indicates that the flow is supporting the realization of an ExtendedActivity. Each instance of a SupportFlow whose source is not a Material Resource, has a “resource role” which can be “responsible for” or “participates in”.
- **FlowNode:** an abstract representation of the diagram’s elements that are connected together by means of flows.

A **FlowNode** is a super class of four other classes:

- **ExtendedActivity:** this represents the functional unit of a process. An Extended activity can be broken down into several activities. In such case, it is called a 'Structural Activity'. An activity that has not been broken down will be called an “Atomic Activity”. In addition, an Extended Activity can be defined as “starting” or “ending» activity, which demonstrates where the process starts and ends.
- **Resource:** an abstract concept representing resources used by a process to support one or several activities. It can be of three types: human, material, and IT.
- **Connector:** used to represent the origin or the destination of a flow when the origin or the destination is outside the current diagram. Possible roles are: process connector, internal connector, and external connector.
- A ProcessConnector class has a reference pointing to a process outside the current diagram.
- **Logical Operator:** this represents a convergence or a divergence of multiple flows. There are four different kinds of logical operators: ConvergingAnd, DivergingAnd, ConvergingOr, and DivergingOr.

Static Semantics (Constraints and Trigger)

Static semantics define the rules (constraints) that govern relations between classes. In Extended Actigram Star language, *constraints* on Flow and the type of its source and target are defined. In order to represent these constraints:

- Textual annotations can be associated with the UML meta-model at design level.
- Object Constraint Language OCL can be used at design and implementation level.

Table bellow (Table 8) summarizes the rules that apply to the utilization of Flow, depending on the target and source nodes to be connected.

A Flow can *trigger* an extended activity, which means that an extended activity will wait for the flow in order to start its mission or to function. This triggering characteristic is governed with constraints depending on the flow’s source, target, and the flow’s type:

- The target of a flow must be an ExtendedActivity.
- The flow must be an OutputInputFlow or a ControlFlow.
- A SupportFlow cannot be a triggering flow.

According to OMG UML 2.0 reference specification document (<http://www.omg.org/cgi-bin/doc?formal/2009-02-02.pdf>), State machines can be used to specify behaviour of various model elements. For example, they can be used to model the behaviour of individual entities (e.g., class instances). The state machine formalism described by the Meta-model of Figure 21 is an object-based variant of Harel statecharts (<http://www.wisdom.weizmann.ac.il/~dharel/SCANNED.PAPERS/Statecharts.pdf>).

BPMN 2.0

It is difficult to present a complete view of the BPMN2.0 meta-model due to its large extent. For this reason, the best is to refer directly to its specification, available at the OMG website: <http://www.omg.org/spec/BPMN/2.0> (OMG 2011b).

However, to help in the analysis performed when defining the mappings with EA* and WSDL, below are included very brief descriptions of the most relevant BPMN 2.0 concepts considered.

User Task

A **User Task** is a typical “workflow” **Task** where a human performer performs the Task with the assistance of a software application and is scheduled through a task list manager of some sort.

Script Task

A **Script Task** is executed by a business process engine. The modeller or implementer defines a script in a language that the engine can interpret. When the **Task** is ready to start, the engine will execute the script. When the script is completed, the **Task** will also be completed.

Service Task

A **Service Task** is a **Task** that uses some sort of service, which could be a Web service or an automated application.

Exclusive Gateway

A diverging **Exclusive Gateway (Decision)** is used to create alternative paths within a **Process flow**. This is basically the “diversion point in the road” for a **Process**. For a given instance of the **Process**, only one of the paths can be taken.

Sub-Process/Call Activity

A **Sub-Process** is an **Activity** that encapsulates a **Process** that is in turn modelled by **Activities**, **Gateways**, **Events**, and **Sequence Flows**. Once a **Sub-Process** is instantiated, its elements behave as in a normal **Process**.

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.

Participant

A **Participant** represents a specific **PartnerEntity** (e.g., a company) and/or a more general **PartnerRole** (e.g., a buyer, seller, or manufacturer) that are Participants in a **Collaboration**. A **Participant** is often responsible for the execution of the **Process** enclosed in a **Pool**; however, a **Pool** MAY be defined without a **Process**.

Message Flow

A **Message Flow** is used to show the flow of **Messages** between two **Participants** that are prepared to send and receive them.

In **Collaboration Diagrams** (the view showing the **Choreography Process** Combined with **Orchestration Processes**), a **Message Flow** can be extended to show the **Message** that is passed from one Participant to another.

WS-BPEL 2.0

OASIS recognizes the need for an independent (and choreographed) representation of the interactions between parties. WS-BPEL provides a language for the specification of Executable and Abstract business processes. By doing so, it extends the Web Services interaction model and enables it to support business transactions.

A BPEL Process is a container where one can declare relationships to external partners, declarations for process data, handlers for various purposes and, most importantly, the activities to be executed. A process definition is made of one activity, and several exist that consume or provide messages to web service partners:

- receive, receiving messages from an external partner
- reply, used in conjunction with the receive activity it allows to return data to the caller
- invoke is used to call a web service provided by a partner.

The process logic can be structured in with several elements such as sequence (sequential order), if-else (conditional branching), while (repetitions), repeatUntil (repetitions), etc.

For more information, refer to the OASIS WS-BPEL specification at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

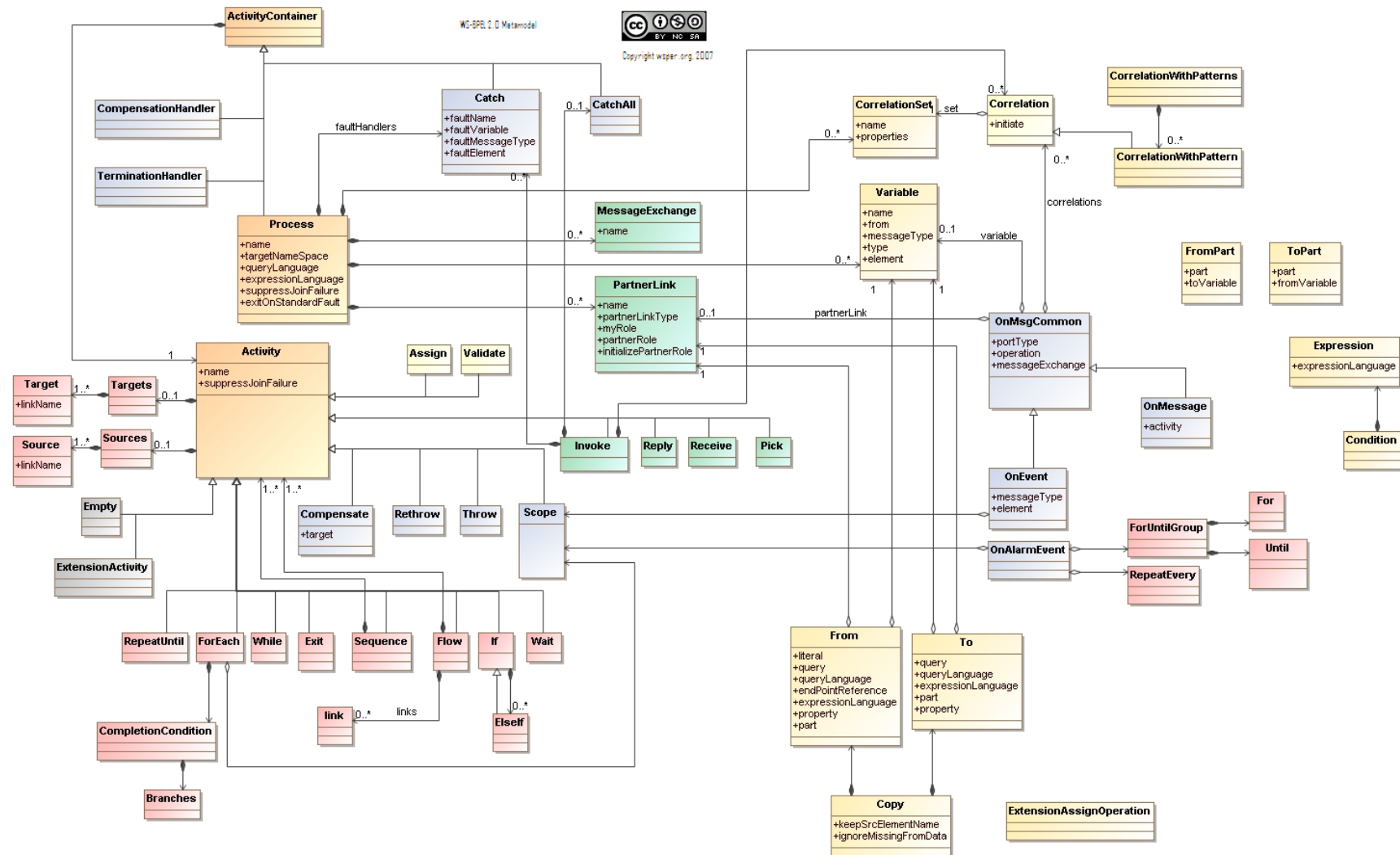


Figure 26: WS-BPEL Meta-Model (from ebPLM(2007))

WSDL 2.0

The WSDL describes services as collections of network endpoints, or ports. The abstract definitions of endpoints and messages/types are separated from their concrete use or instance, allowing the reuse of these definitions.

As illustrated in the meta-model of Figure 27, an endpoint is defined by associating a network address with a reusable binding, and a collection of endpoint defines a service. Messages or types are abstract descriptions of the data being exchanged, and interfaces are abstract collections of supported operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding, where the operations and messages are then bound to a concrete network protocol and message format. In this way, WSDL describes the public interface to the Web service.

The current version of the specification is 2.0; version 1.1 has not been endorsed by the W3C but version 2.0 is a W3C recommendation (see the WSDL specification available at <http://www.w3.org/TR/wsdl20/> for more information).

Table 9: Objects in WSDL 1.1 / WSDL 2.0⁷

WSDL 1.1 Term	WSDL 2.0 Term	Description
Service	Service	Contains a set of system functions that have been exposed to the Web-based protocols.
Port	Endpoint	Defines the address or connection point to a Web service. It is typically represented by a simple HTTP URL string.
Binding	Binding	Specifies the interface and defines the SOAP binding style (RPC/Document) and transport (SOAP Protocol). The binding section also defines the operations.
PortType	Interface	Defines a Web service, the operations that can be performed, and the messages that are used to perform the operation.
Operation	Operation	Defines the SOAP actions and the way the message is encoded, for example, "literal." An operation is like a method or function call in a traditional programming language.
Message	n/a	Typically, a message corresponds to an operation. The message contains the information needed to perform the operation. Each message is made up of one or more logical parts. Each part is associated with a message-typing attribute. The message name attribute provides a unique name among all messages. The part name attribute provides a unique name among all the parts of the enclosing message. Parts are a description of the logical content of a message. In RPC binding, a binding may reference the name of a part in order to specify binding-specific information about the part. A part may represent a parameter in the message; the bindings define the actual meaning of the part. Messages were removed in WSDL 2.0, in which XML schema types for defining bodies of inputs, outputs and faults are referred to simply and directly.
Types	Types	Describes the data. The XML Schema language (also known as XSD) is used (inline or referenced) for this purpose.

⁷ http://en.wikipedia.org/wiki/Web_Services_Description_Language

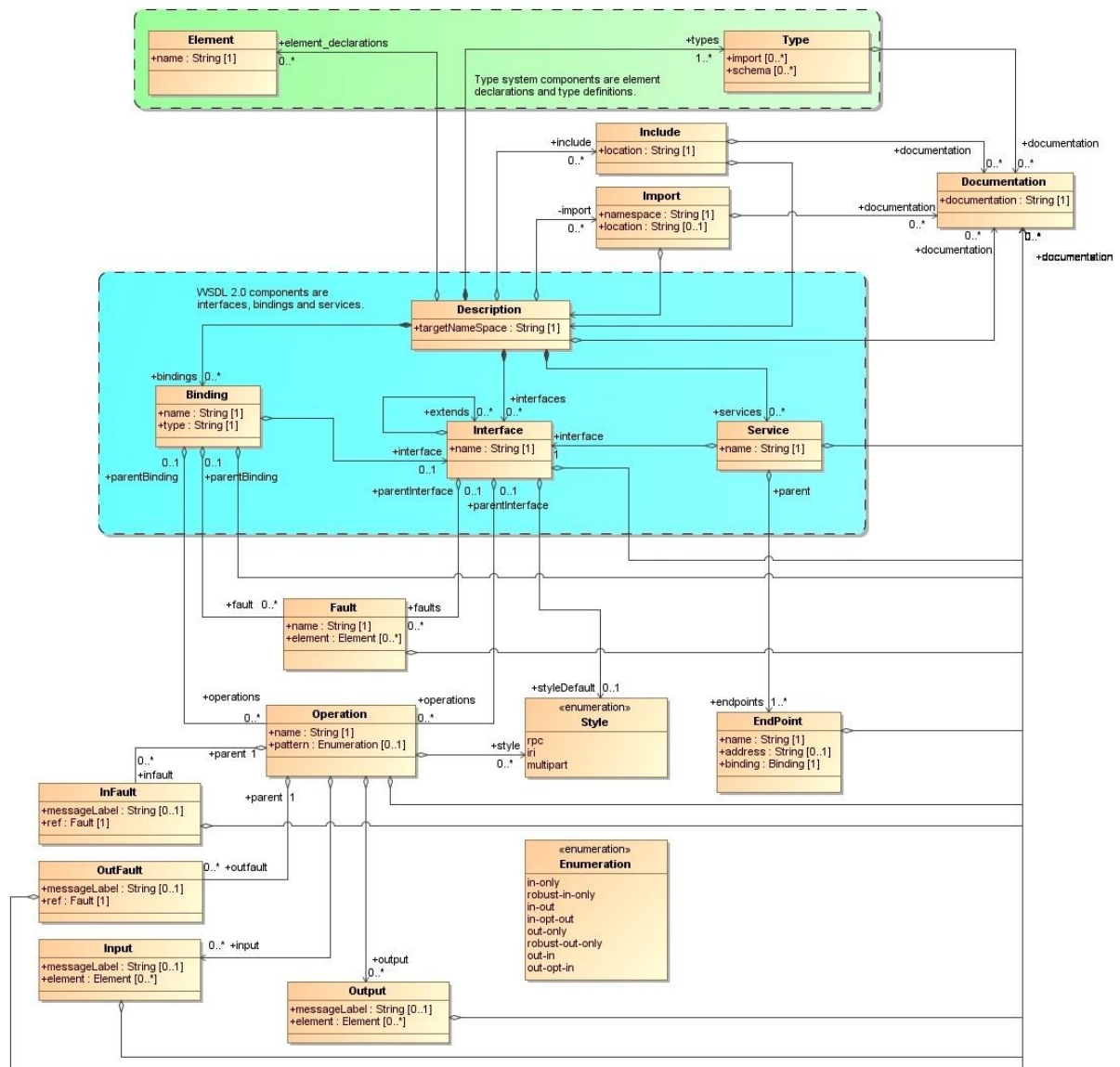



Figure 27: WSDL 2.0 Meta-Model

USDL

Considering a service as an economic or social transaction with a broader context, it is essential to describe the business details as well as the technical details of a service, e.g. the price scheme, the service level agreement, or the terms and conditions when consuming the service and paying for it. As a response to this situation, the Unified Service Description Language (USDL), creates a “commercial envelope” around a service. Technical services may be lifted to business services, but USDL also allows describing more manual or physical services. As many services have a hybrid character with both, a digital and physical or manual footprint, USDL can facilitate the combination and aggregation of such services. Therefore, USDL can be considered one of the foundational technologies to set up an Internet of Services around today’s core enterprise systems.

Project ID 284860	MSEE – Manufacturing Services Ecosystem	
Date: 22/02/2013	Deliverable D11.4 – M15 issue	

For more information, and detailed meta-models refer to the USDL specification at the W3C USDL XG webpage: http://www.w3.org/2005/Incubator/usdl/wiki/Main_Page; or at the Internet of Services: <http://www.internet-of-services.com/index.php?id=264&L=0> .