D11.5
MSEE architecture for Service Modelling: M12 issue

Document Owner:	Gregory Zacharewicz (UB1)
Contributors:	Carlos Agostinho (Uninova), Hassan Bazoun (UB1), Yves Ducq (UB1), Manuel Hirsch (DITF), Tobias Maschler (DITF),
Dissemination:	Public
Contributing to:	WP11
Date:	15.11.2012
Revision:	Version 1.0

VERSION HISTORY

1	DATE	NOTES AND COMMENTS
2	01.11.2012	CREATION OF THE TABLE OF CONTENTS
3	09.11.2012	FIRST INPUT FROM PARTNERS
4	15.11.2012	VERSION 1.0
5		

DELIVERABLE PEER REVIEW SUMMARY

ID	**Comments**	**Addressed (✔) Answered (A)**
1	Exec Summary to be enhanced	✔ Done
2		
3		
4		
5		
6		
7		

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 List of Acronyms and Abbreviations

ATL	Atlas Transformation Language
BPMN	Business Process Modelling Notation
BSM	Business Service Models
CIM	Computation Independent Models
DEVS	Discrete EVent Specification
EMF	Eclipse Modelling Framework
EMOF	Essential Meta Object Facility
G-DEVS	Generalized Discrete EVent Specification
HLA	High-level Architecture
IT	Information Technology
LCIM	Levels of Conceptual Interoperability Model
MDA	Model-Driven Architecture
MDD	Model-Driven Development
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
PIM	Platform Independent Models
PIM4SOA	Platform-independent model for service-oriented architecture
PSL	Process Specification Language
PSM	Platform Specific Models
QVT	Query View Transformation
RTI	Run Time Infrastructure
SLM	Service Lifecycle Management
SOA	Service-Oriented Architecture
SoaML	Service oriented architecture Modelling Language
TIM	Technical/Technology Independent Models
TSM	Technical/Technology Specific Models
UEML	Unified Enterprise Modelling Language
UML	Unified Modelling Language
WSDL	Web Service Definition Language
XMI	XML Metadata Interchange format

2 Executive Summary

This deliverable follows the work of D11.1 and D11.3 in the scope of WP1.1. It is specific on modelling & transformation platform specification and envisages complementing the MDSEA architecture with some transformations strategy, validation and export framework.

The goal of the deliverable is to present and describe some specific parts of the modelling platform from BSM down to the TIM level in addition to the work results provided in previous deliverable D15.2. The goal is to emphasise some facilities proposed to the users and the interoperability with other systems. It does not focus on the Modelling techniques and model transformation already specified in D11.1 and D11.3.

T1.1.5 outputs are elements for the platform architecture specification that will support the SLM tools.

In MSEE, conceptual reference ontologies of service domain have been created or have been reused from exiting ontological libraries (e.g. D23.3 and 23.4). Several ontologies can coexist to describe the same concepts, a wide spectrum of approaches for the harmonization of the various ontologies that represent the same concept has been proposed. This deliverable proposes to use the MENTOR methodology to match the ontologies. In addition the specification of the architecture will provide a description language to support the transformation of conceptual ontology to technical ones. It will give an ontological support for the model transformation in the platform. The ontologies of D23.3 and D23.4 are the reference ontologies (generated using a similar methodology), this deliverable envisages complementing those (according to MENTOR) with Mediator ontologies to assist the transformation process and keep traceability of mappings

In addition, it introduces a bridge with USDL in order to gain in interoperability with standards of the domain. Finally it introduces a process for model validation by simulation using a REST based communication with models checking server. At PIM level, it will support the composition and orchestration of services made available through the platform library (USDL and WSDL) of reusable and interoperable existing models. It will draw the specification for the platform to provide a semantic support for the definition, creation, editing and management of the service models

Finally the specification of the platform will propose wizards to instantiate quickly service interface based REST state description to allow taking into account model user history of access in the objective of validating models.

3 Introduction

The work package 1.1 is working at specifying the Modelling platform that will automate at most the models transformation of from conceptual level to more technical levels. Assuming the MDA/MDI principles, this task will complement and adapt in practice the Modelling transformation concepts of WP1.1, WP1.3 to specify the SLM Modelling tool and the new collaborative architecture for business-IT interaction and distributed decision making in virtual factories and enterprises. The core technical description of the Modelling platform was the subject of the D15.2, D23.3 and D41.1 deliverable released previously. The prescriptions provided in this deliverable envisage the connection and use of ontology, USDL and REST in the platform and try to discuss unaddressed questions in the already provided specification that can be proposed for implementation in the V2 release of the Modelling platform.

At TIM level, the platform needs to be specified regarding the functionality it will propose and anticipate on the techniques it will use. The idea of this deliverable is to describe how some existing practices in the domain of information system can be usefully instantiated in the platform.

On the Modelling aspect, the semantic approach can help the user of the tool to transform and to enrich the model; chapter 5 of the document will describe the benefits to MSDEA service modelling coming from the use of ontology.

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, thus a dedicated deliverable chapter presents the specification proposed to reach that goal.

Finally, the seventh chapter will present the specification to prepare the compatibility with REST that can give support to model verification and validation at the transformation to from TIM to TSM.

4 Recall of Inputs from WP 1.1 & 1.3

This deliverable is mainly linked with the specifications provided in the WP 1.1 & 1.3 that specify the model transformation and the model formalisms selected to be implemented in the platform. For these reasons a recall of these works is useful in this deliverable.

4.1 T1.1.1 Recalls

Task 1.1.1 provided the MSDEA architecture for service modelling identifying languages and concepts required for Business Service Modelling (BSM), Technology Independent Modelling (TIM), and Technology Specific Modelling (TSM). The MSEE servitization methodology details the different steps from the definition of a strategy for future products and services to the modelling of the "to-be" business services (BSM) and the corresponding technical specific models (TSM), where transformations are required to accomplish some of the steps.

The following Figure proposed in the D11.1 shows the proposed structured approach including all the required steps to go from the definition of the objectives of the transition to the detailed definition of all the required resources inside the enterprise and the Service System.

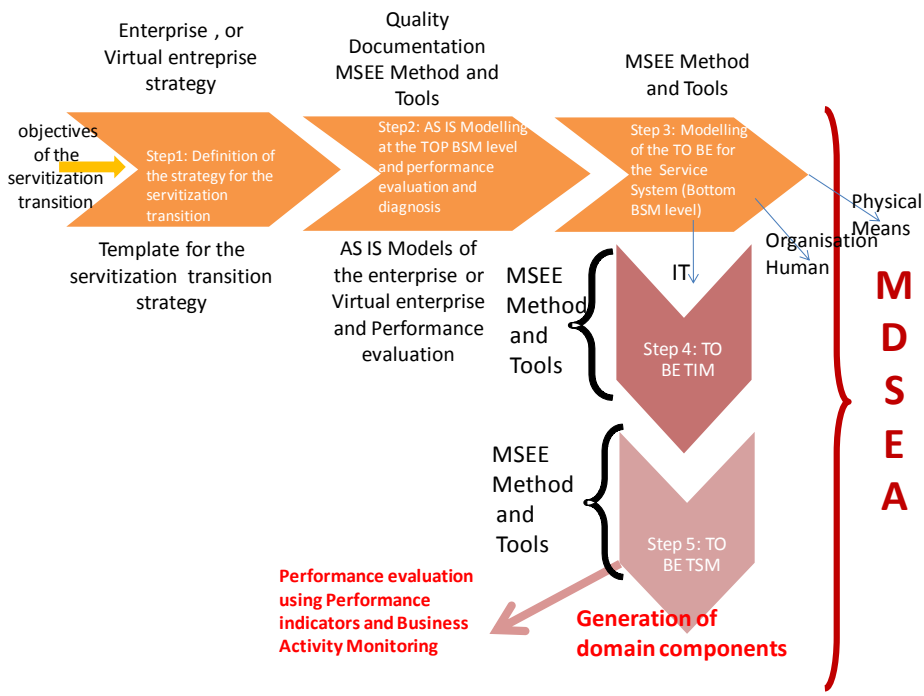

Figure 1: MSEE method using MSDEA tools and concepts

The horizontal sequence initializes the study to reach the TO BE for the Service System.
The vertical sequence allows implementing the MDSEA in order to determine the components of the Service System by domains.

4.2 T1.1.3 Recalls

In task T1.1.3, 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 (vertical transformations), or at the same level to enable sharing and interoperability among

different enterprises (horizontal transformations). Therefore, MDSEA transformations approach applies the distinction between vertical and horizontal transformations, providing interoperability and portability characterized by horizontal transformations among different service systems, at the same degree of relevance as the traceability features of vertical transformations, linking requirements, design, analysis, and testing models of the several MDSEA abstraction levels.

In this context, the MDSEA transformations are specified according to parameters defined along three axes (see Figure 2):

- Axis 1 - Modelling levels, defined according to the reference architecture categorization proposed by OMG, which envisages that real world data is modelled using four levels that go for data itself (M0) to the meta-meta-model (M3);
- Axis 2 - MDSEA levels, which, being inspired on the MDA/MDI enables Service System vertical transformations among the three abstraction levels, i.e. BSM, TIM, and TSM;
- Axis 3- Ecosystem integration, which, starting from a minimum of two systems represents the P2P horizontal transformations among the multitude of systems part of the enterprise service ecosystem. Instead of defining direct transformations among the several enterprise and modelling language specific models, MDSEA envisages that transformation data can go through MDSEA reference formats (derived from the MSEE deliverable D11.1 templates), separating concerns to a neutral format, where the mappings are defined easier.

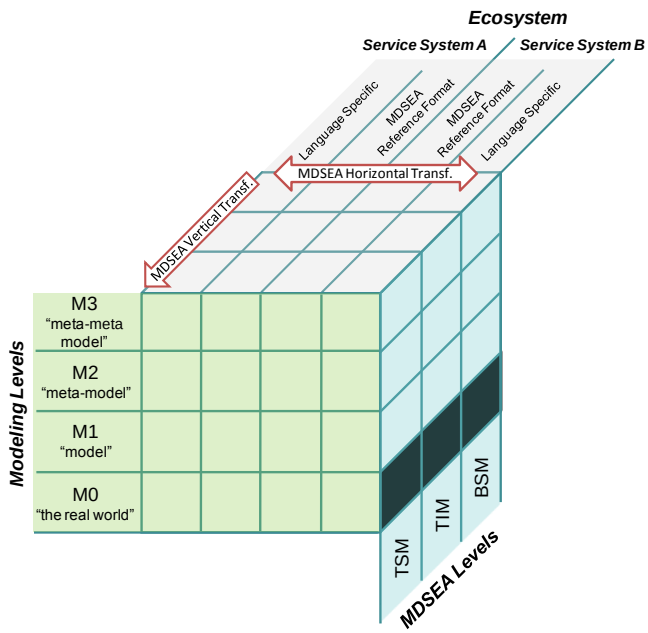

Figure 2: MDSEA Transformations Framework introduced in D11.3

These recalls show that deliverables 11.1 and 11.3 were centred on models and model transformation. Assuming these results we propose in the following to add some facilities to make the models easier to transform with semantics more interoperable with UsDL and some tools for model validation.

5 Use of Ontology for Service Modelling

Ontologies play an important role in order to promote and facilitate interoperability among ICT systems. It is an explicit specification of a conceptualization that refers to the shared understanding of some domain interest that may be used as a unifying framework to facilitate knowledge sharing and interoperability between independently developed subsystems (Gruber 1993) (Hayek 1945). Thus, ontologies allow key concepts and terms relevant to a given domain to be identified and defined in a structure able to express the knowledge of an enterprise. Its recognized capacity to represent knowledge, to facilitate modelling, reasoning, use and exchange of knowledge between systems, contributes to increase computational intelligence.

However, due to the worldwide diversity of enterprises, a large number of ontologies have appeared representing the same segment of reality. Thus, various enterprises, even within the same domain often do not understand each other because of the use of different ontologies and knowledge structures. For that reason, a wide spectrum of approaches for the harmonization of the various ontologies that represent the same concept has been proposed (e.g. MENTOR (Sarraipa 2010)).

Considering the above, also service modelling in the scope of a manufacturing enterprise ecosystem can be complemented with the use and definition of ontologies at the different levels of the MSDEA architecture. In fact, reference domain ontologies at the level of BSM support a common understanding of the domain and business requirements, thus enabling the specification of interoperable services at early stages of modelling. Also, reference technical ontologies at the level of TIM can be useful in the service modelling enabling to share the same understanding upon IT, Human, and Machine resources involved in the manufacturing service definition. Also, the MSDEA transformations framework (specified in Task 1.1.3), which is responsible for the realization of the service through vertical and horizontal transformations envisages to associate semantic knowledge to models and mappings. Once the syntactical correctness has been verified, semantic interpretation, which goes beyond syntax or structure, must be understood and unambiguously defined based on the context of the mapping definition possibly through mediation ontologies that support both types of transformations (for service modelling, vertical transformations are more relevant, relating BSM, TIM and TSM modelling levels).

In conclusion, as illustrated in Figure 3, the service modelling process can benefit from the use of ontologies in two different ways, i.e. the agreement of common semantics for the service modelling enrichment though reference ontologies (right side of the figure), and the support to transformations using traceable mediation ontologies that relate the concepts among source and target concepts in an MSDEA model transformation process (left side).

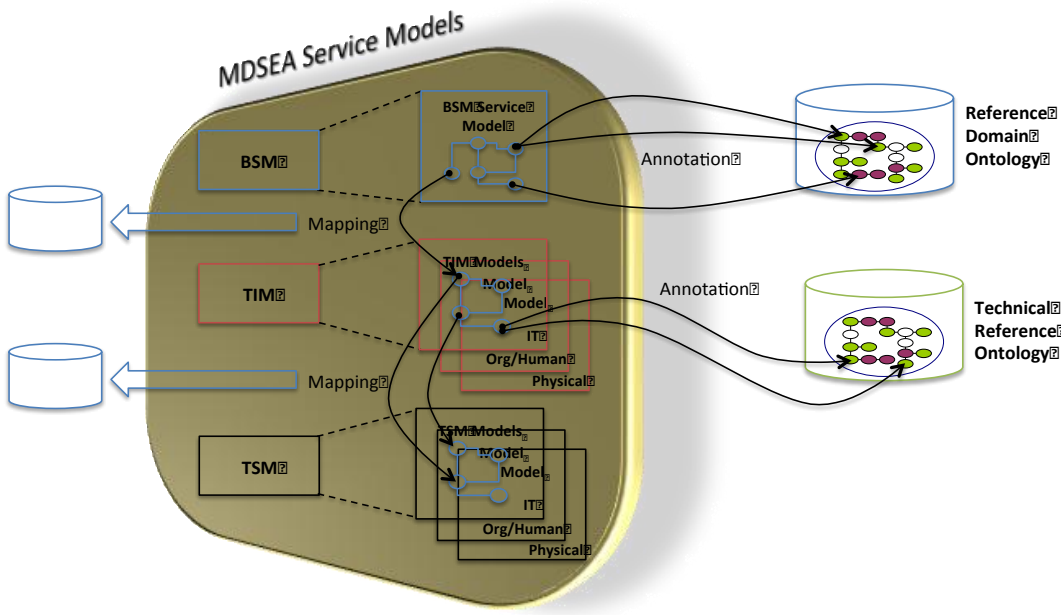

Figure 3: Ontologies in MSDEA service modelling

5.1 Background: The MENTOR Methodology

The "Methodology for Enterprise Reference Ontology Development" – MENTOR is a collaborative methodology for building a reference ontology, as it addresses the objective of achieving shared representation of a domain's knowledge. Since such community reference ontology has to represent all the involved enterprises, the more collaborative this process could be, the easier the community will reach a wider representative ontology. Thus the required characteristics for such reference ontology building would be: ontology building from scratch; ontology reengineering; cooperative building and merge methods, in such way that allows the enterprises enrolled, to keep their own ontologies or semantics unchanged internally.

MENTOR methodology was identified as the only that aggregated all the mentioned characteristics (Sarraipa 2010). The NeOn toolkit also addresses all these mentioned characteristics, and the Protégé tool, is the one that facilitates more the collaborative ontology building through its available plug-ins. However, neither NeOn nor Protégé aggregate all the mentioned characteristics in a single methodology, which could support enterprises to establish a common view on their semantics in a new ontology building process, at the same time as semantic mappings are established between their own knowledge base elements and the new mentioned ontology. Thus, its reuse in communications between old semantic and new semantic systems is not fully facilitated. Consequently, MENTOR demonstrates adequacy to a collaborative building of a common reference ontology, which allows enrolled actors to keep their own semantics unchanged internally, but allowing an outside, general and common semantic view generation. In additional, due to its internal semantics record approach, it also facilitates further updates that could come from any internal alterations.

MENTOR is supported by an ontology system constructed as a hybrid system. Hybrid systems are a special class of knowledge representation systems, which are constituted by two or more subsystems dealing with distinct portions of a Knowledge Base (KB) and specific reasoning procedures (Donini 2010). Its aim is to combine the knowledge described by different formalisms in a semantic interoperable way. This methodology is composed by two phases and each phase has three steps, depicted in the Figure 4.

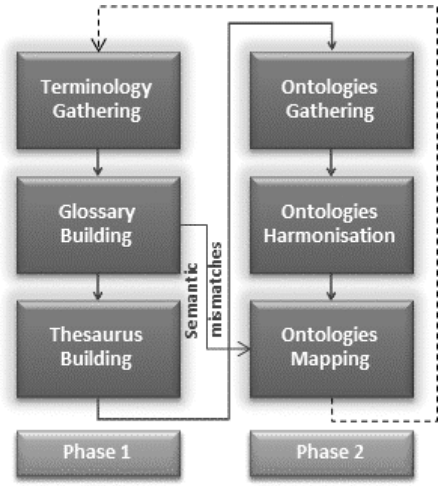

Figure 4: The MENTOR Methodology (light view)

The Lexicon Settlement, or Phase 1, represents the acquisition of knowledge about a domain. This phase is constituted by three steps: Terminology Gathering, Glossary Building, and Thesaurus Building. The first represents the knowledge gathering from all actors. This means that all submit all their terms and definitions on the domain. Then in the Glossary Building step it is established some discussion to define the terms to be the reference and definitions. In this step it is identified semantic mismatches that are recorded in a Mediator knowledge base or ontology. These semantic mismatches records are used in further semantic mappings establishment (see in Figure 1 the arrow that goes from the Glossary Building block to the Ontologies Mapping block). Thus, this step defines the glossary on the domain discussed but also a set of semantic mismatches records in a Mediator knowledge base. The last step of this phase is composed by a cycle where the participants through discussions organised in a Delphi method approach as shown in (Sarraipa 2010), define a taxonomic structure from the glossary terms.

The Reference Ontology Building, or Phase 2, is the phase where the reference ontology is built and the semantic mappings between participant' knowledge representation means (ontologies, concepts, etc.) and the reference ontology are established. This phase is also divided in three steps: Ontologies Gathering, Ontologies Harmonization, and Ontologies mapping. The Ontologies Gathering step realizes ontologies gathering in the domain defined. In Ontologies Harmonization step, is where there are two distinct discussions: one for the ontology taxonomy definition; and other for the properties and rules establishment. At the end of this step is produced the reference ontology of the involved community. The final step of this phase, the Ontology Mapping establishes mappings between each participant's (proprietary) ontology and the reference one. Such process uses the semantic mismatches identified and which was recorded on the Mediation Ontology (MO) as a support for the mapping tables' establishment.

5.2 Ontology Building to Support MSDEA Modelling

This section describes in how MSEE takes up MENTOR methodology and goes beyond, enabling semantic agreement within the process of manufacturing service modelling, promoting enterprise collaboration and interoperability via the services provision.

The process of the development of the MSDEA reference ontologies (domain and technical) is based on the principles of the MENTOR methodology. These principles originally apply to different existing enterprise ontologies and lead to the development of a new one, the so-called "reference ontology". However, within a manufacturing services ecosystem, the MENTOR methodology is adapted to the special features of the exchanged knowledge and

the involved communicating parts. Thus, when an enterprise or any other service provider wishes to submit a new service within the ecosystem, they should provide a set of concepts/terms along with the according definitions that describe and are relevant to each particular service and its domain. In this way, the various candidate services can be defined and further on, identified in the ontology. This process accomplishes the "Terminology Gathering". After the submission of various enterprises services and terminology, it is followed to the development of a "Glossary". Then all the terms and definitions are organised in a classification tree resulting in the establishment of the "Thesaurus".

5.2.1 Adapted MENTOR Methodology

The MENTOR methodology, as it is originally conceived, applies to ontologies, ("O" in left part of Figure 5) and leads to the development of a reference ontology ("$O_{(R)}$" in right part of Figure 5), leaving the initial ontologies unchanged. In MSDEA ontology building, the methodology is not applied to ontologies, but to sets of services terms and their definitions, leading to the creation of a reference ontology ("$O_{(R1)}$" in right part of Figure 5). The main difference between both approaches is related to the input sources: in the first there are ontologies; in the second there are terms able to categorise the introduced services.

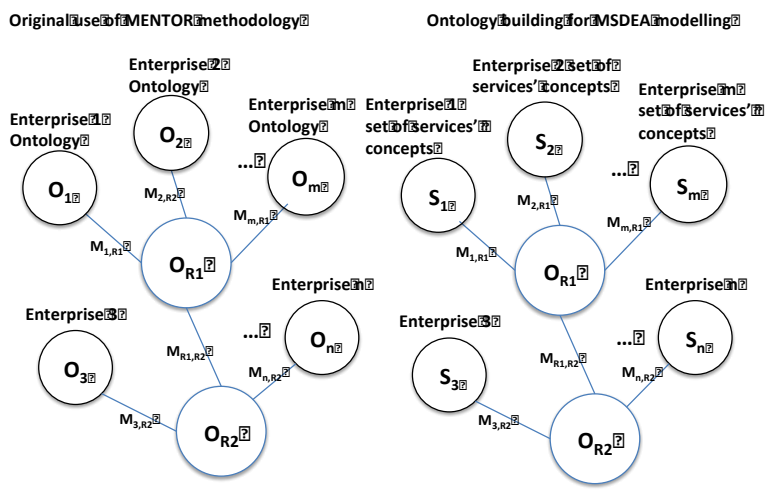

Figure 5: Ontology building for MSDEA Modelling

If enterprises and/or service providers would like to model and develop new services, it is performed a semantic harmonization of all the involved knowledge using the reference ontologies for the BSM and TIM levels. In this case, the semantic harmonization process is implemented merging or mapping (categorizing) the knowledge of the various sets of terms with the existing version of the reference ontology. If conceptual equivalent terms are already present in the reference ontologies, semantic mappings are established between the knowledge of these newly submitted sets of services terms and the knowledge of the existent reference ontology version. If not, such new services terms are introduced as in a regular merging process.

Ontology mapping is the process that relates the vocabulary of two ontologies that share the same domain of discourse. In the present case, the mapping process relates terms using "conceptual" links between the services suppliers and the MSDEA reference ontologies, relating different knowledge views. By such process, common view ontology is built allowing different conceptualisations between services suppliers. This permits an intelligent reasoning over the ontology mainly because it accepts not only the reference ontology terms but also the

others originally introduced by services suppliers because they are recorded through the mentioned mappings on the MO.

The right part of Figure 5 shows how each new version of the Ontology is built and how it evolves by embedding new knowledge when new services are being submitted. This new knowledge feed, leads to a sustainable evolutional learning of the manufacturing services knowledge.

5.3 Ontology based models enrichment

Semantic enrichment of models is performed thanks to the provision of semantic annotations associated with models and components (parts) of models: the main aim of these annotations is to make explicit the meaning of the models. Within the frame of MSDEA this enrichment can be associated with the act of modelling, and as explored in MSEE deliverable D11.3, also with model transformations.

The purpose of annotations is to describe the content of "something" (*annotated object)* and therefore annotations may be considered as *meta-data.* They may be provided under different forms, like links, paths, notes, comments, highlights, etc. Moreover, annotations may appear as *informal* (like a personal margin note while reading a book or an article) or *formal*, meaning that the annotation expressions may be expressed according to given structural standards (like RDF and RDF Schema) with a formal syntax, or other well-founded language (e.g First Order Logic, Description Logic, etc.).

Machine-readability of the of the service model annotation increases according to the degree of annotation formality. This assumes that no implicit assumptions should be made and no ambiguity should persist to enable a common interpretation and understanding of the annotations. Part of this common understanding relies on the use of one or several ontologies.

Therefore, the adopted approach is to procure semantic enrichment of service models thanks to semantic annotations i.e. "tag" models with additional information from reference ontologies to make their meaning more explicit (right side of Figure 3). Being the act or process of furnishing critical commentary or explanatory notes, annotation for semantic enrichment can be represented by graphics and/or textual information attached to a specific model, and can provide the following additional knowledge (Boudjlida 2007) to the original model:

- *Decoration*: textual comments associated with the resource;
- *Linking*: links to other sources of information;
- *Instance Identification*: the annotated object is an instance of a given class within an ontology and the annotation content may be a link to that class;
- *Aboutness*: no assertion is made about the existence of an instance of the concept, but there is a loose association with the concept;
- *Pertinence*: the target of the annotation may be of interest for the annotated object.

5.4 Ontology support to model traceability

The annotation for model traceability can be defined as a specific usage of the annotations for models, linking different modelling structures (e.g. service models), and storing those links on dedicated knowledge repositories (i.e. Mediation Ontologies - MO).

MO aims at storing the information that should be gathered for use in the model traceability process. In service modelling, traceability is represented by a relationship between model entities (mapping) that have changed as consequence of a specific model operation in a backward and forward direction (e.g. BSM to TIM transformation). In this situation, traceability will make possible to follow a model evolution.

A possible scenario (Figure 6) is when a model experienced some evolutions (e.g. Model Operation N; Model Operation N+1). The "Entity δ" is the result of "Entity α" in Model N-1, applied to a Model Operation N-1 (Entity δ= Model Operation N (Entity α)). As consequence "Entity x" is the result of "Entity δ" in Model N, applied to a Model Operation N (Entity x = Model Operation N (Entity δ)). Operations N and N+1 have Model Traceability Instance represented in the MO.

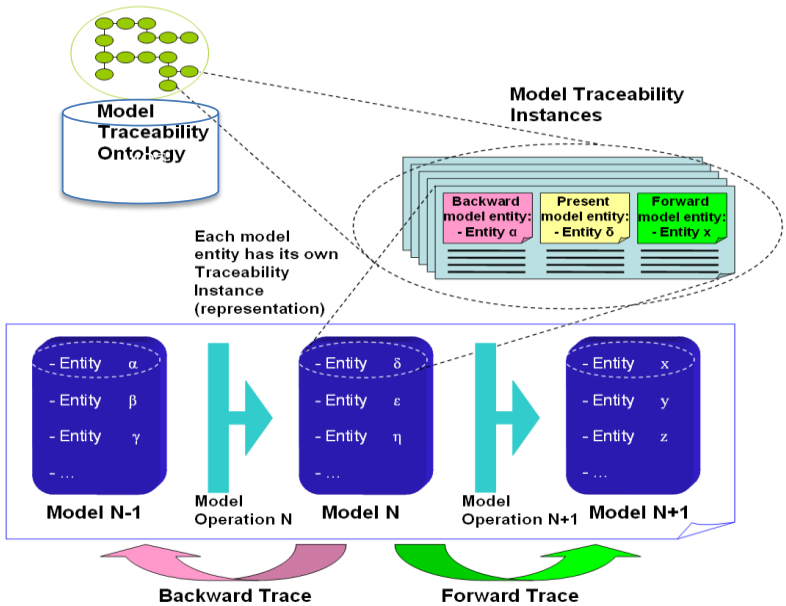

Figure 6: Model Traceability Scenario

Web services languages, e.g. WSDL and BPEL, mostly target the description of technical characteristics of services. Nevertheless they do not cover the more general scope of business services (BSM level) and do not support solutions for this conceptual consideration. Business services have a much coarser grain than typical Web services, mainly because they are concerned with the end-to-end delivery of an added value and outcome. Ultimately, they may be realized by a technical service, but there is definitely the need for more than the technical description of a Web service interface.

More comprehensive service Ontologies (presented in the previous section) suffer from the same problem. Project like OWL-S and WSMO are closely related to technical Web Service standards. At the opposite, business-focused Ontologies, e.g. E3Service ontology, PAS 1018, are dedicated to concepts. Some specifications try to bridge the gap in domain-specific industry standards context, e.g. ebXML. Even so, while they do cover technical and business aspects in a specific context, they lack generalization.

In order to overcome the gap between the business and the technical perspectives and provide what can be considered as an important building block of the Internet of Services (both for IT Services as well as Manufacturing Services), SAP Research has proposed a new conceptual model to describe services. The language is called Unified Service Description Language (USDL) (W3C USDL 2010). USDL is a platform-neutral language for describing services. It was consolidated concerning services-related research as an enabler for wide leverage of services on the Internet. The kinds of services targeted for coverage through USDL include: purely human/professional, transactional, informational, software component, digital media, platform and infrastructure. USDL on a whole is made up of a set of modules, each addressing different aspects of the overall service description. Modularization was introduced to improve readability of the model, which drastically grew in size compared to its predecessor. The modules have dependencies among each other, as they may reuse concepts from other modules. USDL has been involved in the following EU DG INFSO projects: FAST, RESERVOIR, MASTER, ServFace, SHAPE, SLA@SOI, SOA4ALL, FI-WARE. Nevertheless this language is not yet standardized and not fully adopted in the business domain.

In the proposed platform the modelling at BSM level is tackled by GRAI Extended Actigram and GRID. Nevertheless the USDL is now recognized in the domain, starting from SAP works it is now under discussion of W3C standardisation.

6.1 USDL recalls

6.1.1 USDL Origin

The Unified Service Description Language (USDL) is a platform-neutral language for describing services The Unified Service Description Language (USDL) is proposed as a “master data model for services” to describe various types of services ranging from professional to electronic services. It aims at a holistic service description putting a special focus on business aspects such as ownership and provisioning, release stages in a service network, composition and bundling, pricing and legal aspects among others, in addition to technical aspects (W3C USDL 2010) (ref Figure 7). It was initially consolidated from SAP Research projects (USDL site 2012) concerning services-related research as an enabler for wide leverage of services on the Internet. SAP Research believes that with the rise of commoditized, on-demand services, the stage is set for the acceleration of and access to

services on an Internet scale. It is provided by major investments through public co-funded projects, under the Internet of Services theme, where services from various domains including cloud computing, service marketplaces and business networks, have been investigated for access, repurposing and trading in large settings.

6.1.2 USDL focus

The kinds of services targeted for coverage through USDL include: purely human/professional (e.g. project management and consultancy), transactional (e.g. purchase order requisition), informational (e.g. spatial and demography look-ups), software component (e.g. software widgets for download), digital media (e.g. video & audio clips), platform (e.g. middleware services such as message store-forward) and infrastructure (e.g. CPU and storage services).

Figure 7: USDL position

A generic service description language – like USDL – acting as a "one size fits all" for domains as diverse and complex as banking/financials, healthcare, manufacturing and supply chains, is difficult to use and therefore not sufficient. First of all, not all aspects of USDL apply to all domains. Rather, USDL needs to be configured for the particular needs of applications where some concepts are removed or adapted while new and unforeseen ones are introduced. A particular consideration of this is allowing specialized, domain-specific classifications such as those available through vertical industry standards to be leveraged through USDL. In addition to this, the way in which USDL is applied for deployment considerations, e.g., the way lifecycle versioning applies, needs to be managed without compromising the fundamental concepts of USDL. In other words, USDL needs to be applied through a framework which allows separation of concerns for how it is applied and tailored to concrete applications. This need has led to the USDL framework where the concepts of the USDL meta-model as a core are specialized through the USDL Application meta-model. A non-normative specialization of the USDL meta-model with the USDL framework is provided to illustrate how a service directory of a specific Service Delivery Framework (proposed by SAP Research) can be conceptualized through USDL. In this way, an insight is available for an application of USDL and its USDL Application meta-model.

6.2 Rationale for USDL use

There exists a plethora of existing service description efforts that can be grouped into different strands. Each of the strands has its own motivation and representation needs for capturing service information. The individual efforts can be attributed to the following criteria:

(i) whether the scope of the effort lies in capturing IT or business aspects of services or the whole service system.

(ii) the purpose of the corresponding effort, e.g., enabling of normative data exchange, facilitation of software engineering, or acting as reference model.

(iii) whether the effort is able to capture business network relationships between services.

(iv) whether the effort is standardized. Table 1 collapses the individual efforts of each strand and lets us conclude that USDL is the only effort which covers IT and Business aspects, serves both a reference and exchange purpose, considers business network related information and Is about to be standardized.

Table 1: Comparison of USDL regarding other strands

Strand	Scope	Purpose	Business network	Standardized
SOA efforts	IT	Exchange or Engineering or Reference	No	Various
Semantic Web Service efforts	IT	Automation or Reference	No	No
SaaS efforts	IT	Engineering or Exchange	No	Yes and No
Economic efforts	Business	Reference	No	Yes and No
Service Network efforts	Business	Configuration or Optimization	Yes	No
Service System efforts	System	Reference	No	No
USDL	IT and Business	Reference and Exchange	Yes	Yes

The first strand of service description efforts is the field of Service-oriented Architectures (SOA). SOA is a way of thinking about IT assets as service components, i.e., functions in a large application are factorized in stand-alone services that can be accessed separately. Because of their IT focus, most approaches limit their attention to the field of software architecture. Originally, several standards bodies specified roughly two dozens of different aspects which are collectively known as WS-* (incl. WSDL, WS-Policy, Ws-Security, etc.). Since one of the key components of a SOA is a service registry, the OASIS standards body introduced the concept of Universal Description, Discovery and Integration (UDDI), i.e., a specification for a platform-independent registry. UDDI services shall be discovered according to information such as address, contact, known identifiers, or industrial categorizations based on standard taxonomies. However, UDDI does hardly prescribe any schema for such information. As the concept of SOA matured, calls for support in software and service engineering increased. Hence, the OMG standards body dedicated its focus to software engineering for SOA, and, subsequently defined the Service-oriented architecture Modelling Language (SoaML). Finally, the multitude of description efforts and different definitions of SOA led to a Reference Model for Service Oriented Architecture (SOA-RM) from OASIS. Similarly, The Open Group drafts an alternative reference model in form of ontology for Service-Oriented Architectures (SOA Ontology).

A second strand consists mainly of ontologies in the field of Semantic Web Services. The main goal of Semantic Web Services approaches is automation of discovery, composition, and invocation of services in a SOA by ontology reasoners and planning algorithms. The most prominent efforts are OWL-S and WSMO. Many similar efforts have surfaced in literature. With the many approaches around came the need to specify a reference model for semantic SOAs. Consequently, the OASIS is also about to specify a Reference Ontology for Semantic Service Oriented Architectures (RO-SOA).

The third strand is rooted in the rise of on-demand applications that led to the notion of software-as-a-service (SaaS), covering software applications (e.g., CRM on-demand) and

business process outsourcing (e.g., gross-to-payroll processing, insurance claims processing) to cloud and platform services. The emphasis of service here implies that the consumer gets the designated functionality he/she requested together with hosting through a pay-per-use model. Thus, software-as-a-service is not synonymous with SOA. This difference triggered the Software-as-a-Service Description Language (SaaS-DL). SaaS-DL builds on WS-* to capture SaaS specificities in order to support model-driven engineering. The strand of SaaS also contains a standard, namely, the W3C recommendation called SML (Service Modelling Language). One anticipated use for SML is to define a consistent way to express how computer networks, applications, servers, and other IT resources are described or modeled so businesses can more easily manage the services that are built on these resources.

The fourth strand is driven by schools of business administration and focuses on capturing the purely economic aspects of services regardless of their nature (with less or no focus on IT services and software architectures). The German standard DIN PAS 1018 essentially prescribes a form for the description of services for tendering. The structure is specified in a non-machine-readable way by introducing mandatory and optional, non-functional attributes specified in natural language, such as, classification, resources, location, etc. The PhD thesis of (O'Sullivan) adopts a wider scope and contributes a domain independent taxonomy that is capable of representing the non-functional properties of conventional, electronic and web services. The work compiles the non-functional properties into a series of 80 conceptual models that are categorized according to availability (both temporal and locative), payment, price, discounts, obligations, rights, penalties, trust, security, and quality.

The fifth strand is also economic but draws attention mainly to describing Service Networks, i.e., the ecosystem and value chain relationships between services of economic value. The e3Service ontology models services from the perspective of the user needs. This offers constructs for service marketing, but in a computational way, such that automated reasoning support can be developed to match consumer needs with IT-services. The main focus of this work is to generate service bundles under the consideration of customer needs. The Service Network Notation (SNN) captures similar aspects to the e³Service ontology. However, SNN is an UML model that can be analyzed for measurements of added value for each single participant as well as for the whole network optimization of value flows.

Finally, there are overarching efforts that concentrate on the bigger picture of service systems or service science also taking into account socio-economic aspects. Stephen Alter was one of the first to realize that the concept of a service system is not well articulated in the service literature. Therefore, he contributes three informal frameworks as a first attempt to define the fundamentals of service systems. The work of Ferrario and Guarino (Ferrario 2012) can be seen as a continuation and formalization of Alter's approach (Alter 2008). Although differing in its main notions, they present reference ontology for ontological foundations of service science which is founded on the basic principles of ontological analysis. In turn, this reference ontology forms the core part of the TEXO Service Ontology which extends it by ontology modules for pricing, legal, innovation, or rating information.

6.3 USDL in use in MSEE

From the MSEE project view, USDL should be able to access to MSEE modelling tool database or model repository to get some information relevant to USDL. In other words,

modelling tool should be able to send / inject some information (for example, partner name, address, service functions etc.) into USDL database.

6.3.1 Opening the platform to USDL interoperability

The access to platform resources should be available for the building of USDL structures that are dedicated to business domain.

The Modelling *languages* for SLM methods on BSM, TIM, and TSM level differ in their intended usage from USDL Modelling language. Regarding the Modelling constructs in these languages, the Modelling *constructs* from the BSM/TIM/TSM models of the SLM approach are used to model the service system with respect to different Modelling perspectives during all stages of the SLM service lifecycle.

For the level of models, the 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.

With respect to USDL's usage in the SLM lifecycle, this stage "Service Operation" may be subdivided into sub-stages as proposed here:

1. *Service description* stage: Service offer has been completed
2. *Service discovery and matchmaking stage*: Service offer has been published
3. *Service usage stage*: a service instance has been started, it is currently performed and the service experience is consumed by its service consumer(s).

Within MSEE the main purpose of USDL models is to model a set of selected service aspects that are relevant for sub stages 1 and 2, namely for activities like service description, service discovery, service composition or aggregation, and service consumption / service participant interaction. While some USDL model data can be used within sub-stage 3, this is not the primary focus of USDL support in MSEE.

Projection: The Modelling concepts from BSM, TIM, and TSM model can be projected onto Modelling concepts in the USDL Modelling language. Modelling information from the BSM model, the TIM model, and the TSM model can be mapped into USDL models. Not all information from the BSM/TIM/TSM models is projected into USDL, and USDL can contain additional information not captured in the BSM/TIM/TSM models.

6.3.2 Platform format and USDL matching points

The platform is offering to users 2 main levels for modelling: BSM and TIM. The proposition is to define a matching between the EA* and BPMN models to USDL (Figure 6).

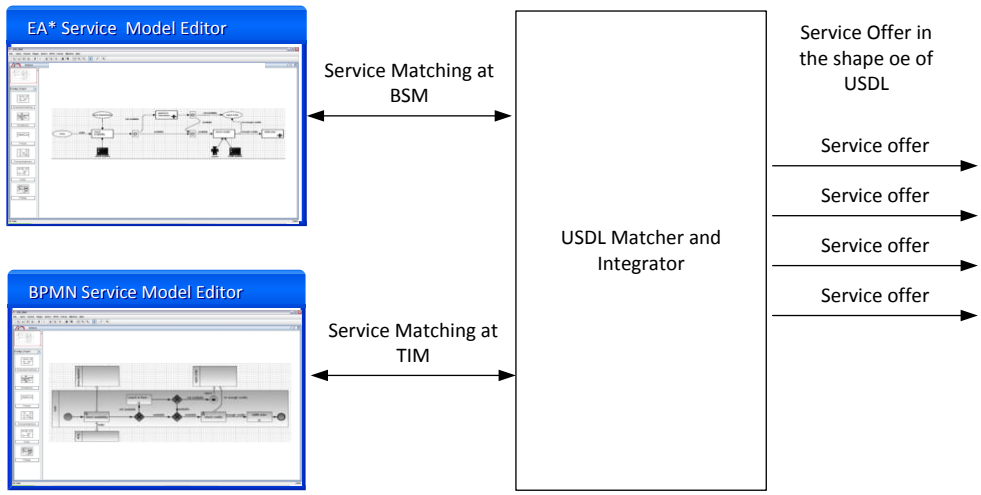

Figure 8: USDL Matching

As an example, Modelling information regarding *participant information* in a service system can be mapped from BSM/TIM/TSM models onto a dedicated USDL module "participants" that is used to model all service participant information in USDL.

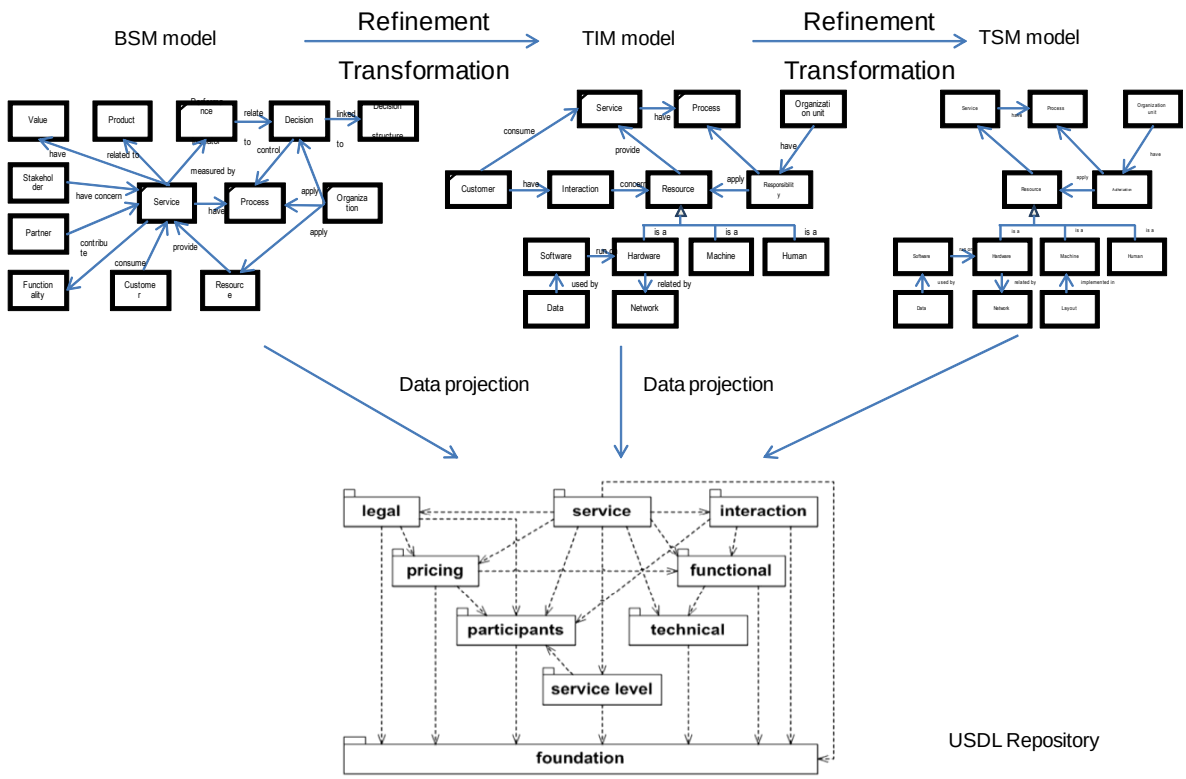

Figure 9: Projection of relevant service concepts onto USDL modeling concepts

The figure 6 introduced in the deliverable 11.1 gives an overview of the Modelling constructs in the USDL module "participants[1]. Organizations and persons can be modeled with a set of attributes (here: Organization, Person). Most prominently, *roles* can be modeled that are enacted by organizations and persons as their agents (here: Role, Agent). Within USDL, a set of pre-defined roles is modeled (Business Owner, Provider, Intermediary, Stakeholder, and Consumer). Other Modelling entities (here: NetworkedProvisionedEntity) can refer to such

[1] http://www.internet-of-services.com, http://www.internet-of-services.com/index.php?id=570, URL accessed on 12.01.2012

roles and can also refer to consumers explicitly (here: Target Consumers). The modelled relationships between these USDL Modelling elements are also shown in the deliverable D11.1.

6.3.3 Detailed Matching

The goal of this section is describe the matching of the MSEE concepts BSM EA* and TIM BPMN into USDL concepts. These concepts are presented in the USDL 3.0 M5 Specification Archive (USDL 2012). The USDL matching is described in the Figure 8.

The Figure 10 introduces the main concepts proposed y USDL.

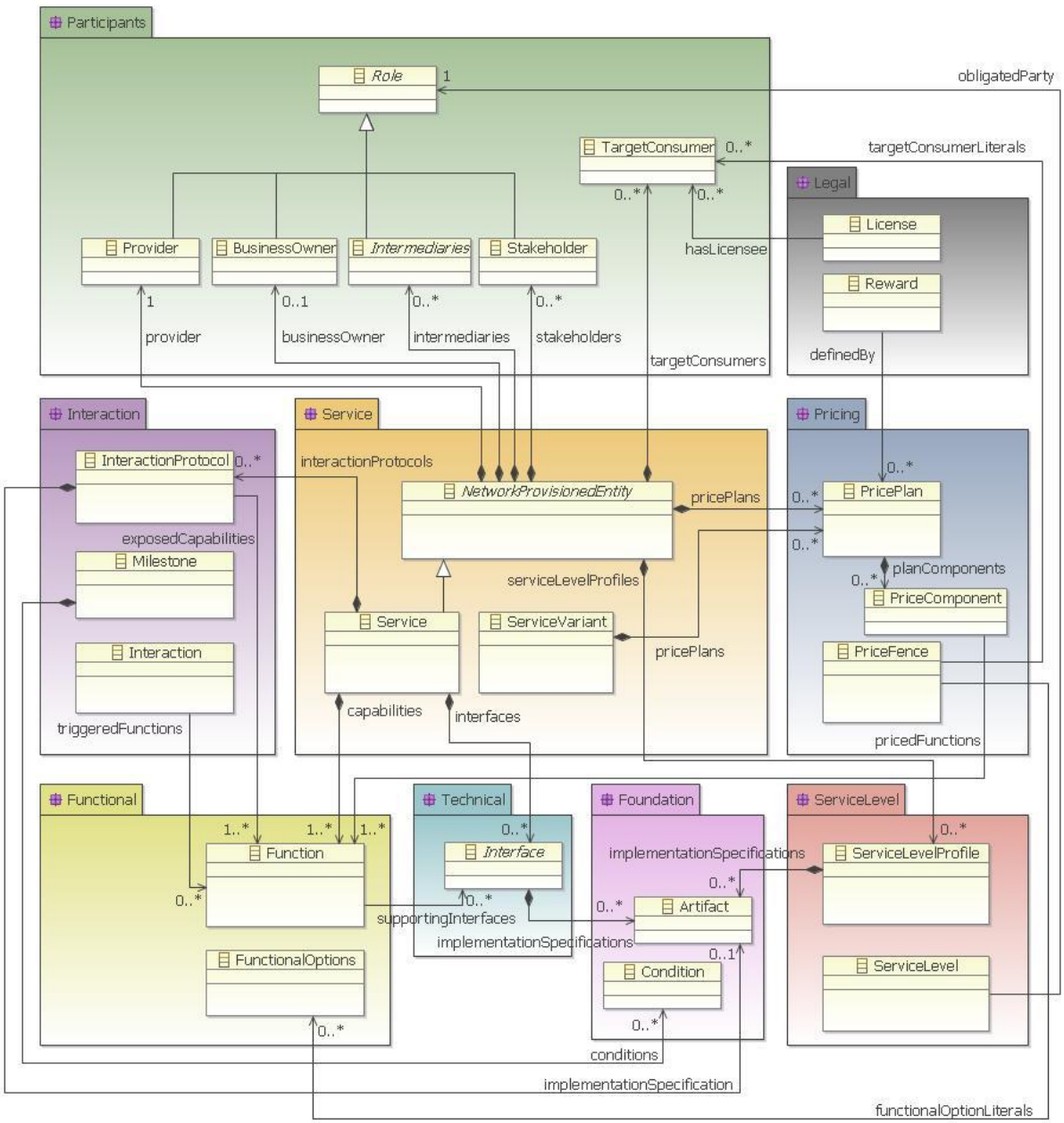

Figure 10: USDL Core concepts

Table 2: Matching of USDL regarding EA* and BPMN

EA*	Condition		BPMN2.0		USDL
Model			Definitions		Elements
Process			Pool, Process, and Participant		Service
Extended Activity	Structural		Activity	Sub Process	Service Variant
	Atomic	It is supported by Human		UserTask	Interaction Protocol
		Supported by IT (no human)		ServiceTask	Artifact
		Only supported by material		Task	Function
LogicalOperator	DivergingOr		Gateway	Diverging Exclusive	Milestone
	ConvergingOr			Converging Exclusive	Milestone
	DivergingAnd			Parallel	Milestone
	ConvergingAnd			Parallel	Milestone
Resource	Material		Data Object		Role
	Human		Performer in a "UserTask"		Role
	IT		Resource (added to list of task resources)		Technical
Flow	Control	If the source is an ExternalConnector or InternalConnector and target is an "atomic" ExtendedActivity	MessageFlow		Interaction
		If the source is an ExternalConnector or InternalConnector and target is a "structural" ExtendedActivity	Catching Message Event, Message flow, and Sequence Flow		Interaction
		If the source is a ProcessConnector or ExtendedActivity	DataObject, and associations		Interaction
	OutputInput	If the source is an ExternalConnector or InternalConnector (and target is an atomic Extended Activity)	MessageFlow		Interaction
		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		Interaction
		If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity)	SequenceFlow		Interaction
		If the source is a structural ExtendedActivity (and target is an ExternalConnector or InternalConnector)	Throwing Message Event, Message Flow, Sequence Flow		Interaction
		If source is an atomic ExtendedActivity (and target is an ExternalConnector or InternalConnector)	MessageFlow		Interaction
	Support	If source is a Material resource	association		Interface
Connectors	External		Participant (Pool)		Participant
	ProcessConnector		Call Activity		Function
	InternalConnector		Participant(Pool) (Black BOX)		Provider, BusinessOwner, Intermediaries, Stakeholder
Dedicated variable			Dedicated variable		Pricing
Dedicated variable			Dedicated variable		Legal

This chapter has introduced the matching between BSM and TIM levels formalisms to USDL concepts. The data structure of USDL can be mostly directly matched with the BPMN and EA* concepts that were introduced for modelling at BSM and TIM.

Nevertheless, In particular it has permit to point the fact that some USDL concepts find no matching in the EA* or BPMN models. Dedicated variables should be created for that objective kin the platform to prepare the interoperability of the models with USDL structure.

7 Service Model behavioural validation with REST & DEVS Simulation

The lower level to be tackled by the modelling platform discussed in this deliverable is the bottom TIM where the platform is preparing the model information for the TSM level. Some model checking and validation can be interesting before going to implementation of the service. Syntactical validation can be quite simple to set up but dynamical ones that depend on time and models states are much difficult. This section is giving the description of state models and REST client states that can be required for that goal. It contributes on this aspect by presenting the interest of local state based representation for the model dynamic validation. In the next sub section a recall is given for REST concepts.

7.1 REST recall

Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. The term REST was coined by Roy Fielding in his PhD thesis[2]. REST consists of a number of architectural style constraints, architectural elements and architectural views. Its main application area is distributed hypermedia applications. In contrast to its counterpart, the architectural style SOAP[3], REST permits loose coupling between services and does not require XML.

The REST architectural style was developed in parallel with the HTTP/1.1 protocol, based on the existing design of HTTP/1.0. It places the emphasis on using a representation of an URL pointed resource to hold the state of an application. A smart REST application uses a small number of verbs, and typically, sophisticated representations (XML being an obvious one). In the future, a good REST application will exhibit key network features such as transparent cache, which can lead to improved performance is widely distributed systems, scalability, reduced server load, statelessness on the server making failover easy, etc. Sophisticated REST applications, (as JAX-RS, RIP, RESTlet) still require careful design in order to make them stateless, which still preserving application semantics, security etc. REST does not prescribe the HTTP verbs, but REST applications can be built using HTTP if carefully done. At this point, a good use of REST in a SOA world would be for simple services, such as a transformation service, PO submission service, etc. where caching, scalability etc.

The emphasis in REST being held in the representation of client side states places it in direct opposition to most web development frameworks. For example cookies and server-side sessions are anathema to REST applications. To avoid these limitations, the W3C Technical Architecture Group (TAG) is investigating the concept of state in client web applications, and there is definitely a REST contingent that thinks cookies have their place. More generally speaking there are debates about how state can be handled well in practical applications using REST, but conventions and shared perception haven't yet emerged. This can be one challenge of the MSEE project for the complex services design and test. For instance the state is a core concept in distributed simulation; it can be reused in this context. Nevertheless, REST can be considered as an abstract architecture needs to be coupled to other technologies like for instance USDL, WSDL or ontology presented earlier or either distributed simulation techniques.

[2] Fielding (2000)

[3] See W3C: SOAP.

An example REST implementation is e.g. the "Common management information protocol" (CMIP)[4]. In the following more details are given to understand REST concepts.

7.1.1 Architectural Style Constraints

Fielding describes it as a set of architectural style constraints[5]:-

- **Client-Server**
  The user interface concerns on *client* side are separated from the data storage concerns on the *server*.
  - Benefits
    - Improved user interface portability across multiple platforms.
    - Scalability improvements by simplifying the server components.
    - Independent life cycles of server and client components.
- **Statelessness**
  The *client-stateless-server* communication style requires that each request from the client to the server needs to contain all necessary information for the server without taking advantage of any stored context in the server. This principle introduces the properties *visibility*, *reliability*, and *scalability*.
  - Benefits
    - *Visibility* is improved because monitoring tasks can derive all necessary information about the request nature from the request's contents.
    - Improved *reliability* as the correction of deficits that lead to (partial) failures can be carried out much easier.
    - As not having to store states between requests, the server can quickly free resources. This improves therefore *scalability* and implementation.
    - The *application state* is entirely kept on client side.
  - Drawbacks
    - The session state is hence kept entirely on the client.
    - Decreased network performance as more repetitive data[6] needs to be communicated, because data cannot be left on the server in a shared context.
    - As the entire application state is kept on client side, the server has much less control over consistent application behaviour[7].
- **Cache**
  Caching constraints improve the network efficiency of the client-stateless-server style.

---

[4] See e.g. ITU (2006).

[5] See Fielding (2000), Chapter 5. The constraints need to be met when developing a client server application's architecture.

[6] Fielding calls this „per interaction overhead"

[7] The semantics for the client-server communication need to be well-defined and correctly implemented within the entire client's life cycle.

These constraints require that the data within a response needs to be labelled cacheable[8] or non-cacheable in an implicit or explicit manner.

- Benefits
  - Some interactions may be eliminated partially or completely. This improves efficiency and user-perceived performance as well as scalability by reducing the average interaction latency.
- Drawbacks
  - Caching may decrease reliability in terms of "aged" data that differed significantly from the one that a current request might have obtained from the server.

- **Uniform Interface**

  REST relies on a general, *uniform interface* between all components. This uniform interface decouples different implementations from the services that these provide. Information between all components needs to be exchanged in standardised forms. The uniform interface needs to comply to the constraints that will be explained later on in the section about architectural elements of REST.
  - Benefits
    - Simpler overall system architecture.
    - Improved visibility of interactions as there are only normalised communication "standards".
    - Increased independency of the implementations of the components within the overall client-server application. This eases the evolvement of each component.
  - Drawbacks
    - Uniform interfaces my decrease efficiency in comparatively specific contexts. REST interfacing is optimised for common communication cases; special forms of architectural interaction may need distinct communication architectures.
- **Layered System**

  The *layered system style* allows composing architectures with hierarchical layers. Components are constrained to interact only with "their" layers.
  - Benefits
    - Layers may be used to encapsulate legacy services or to protect new services from legacy clients. Infrequently used functionality may be moved to e.g. a shared intermediary; this simplifies components.
    - Intermediaries may improve system scalability by e.g. load balancing services across multiple networks and processors or by shared caches.

[8] This means that by marking response data as "cacheable", the server gives the client permission to reuse that response data to quickly answer later, equivalent requests.

Further benefits may be found by implementing security policies via special intermediaries[9].

- Intermediaries may actively transform messages as these messages are self-descriptive and have visible semantics[10].

- Drawbacks
  - Added overhead to data processing.
  - More latency reduces user-perceived performance.

- **Code-on-Demand**
  REST allows that client functionality may be extended by downloading and executing code. As this constraint reduces visibility, it is only an optional one in REST.
  - Benefits
    - To make modules available on request reduces the number of features that need to be implemented at the very beginning of application development.
    - By making features available on request, system extendibility is improved.

7.1.2 Architectural elements

REST itself is an abstraction of architectural elements within distributed (hypermedia) systems. REST itself focuses on the roles of components and their interaction. Basically, it defines the basis of web architectures and therefore of the behaviour of network-based applications. REST relies on the following data elements, to be explained later on:

Data Element	Modern Web Examples
resource identifier	URL, URN
resource	the intended conceptual target of a hypertext reference
resource metadata	source link, alternates, vary
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
control data	if-modified-since, cache-control

Figure 11: REST data elements.

REST components communicate by transferring resource representations encoded in standard data types, selected dynamically by the recipient's capabilities and desires and the resource's nature.

Resources form the key abstraction in REST, as any nameable information can be a resource, e.g. documents, images, services, collections of other resources or even real objects. Therefore, a resource is a conceptual mapping to a set of entities, but not the entity that corresponds to the mapping at any particular point of time. A resource may as well map to an

[9] E.g. firewalls.

[10] provided as uniform interfaces

empty set; this permits to establish references – even before the addressed entities are available. This approach that clearly separates the resource from its instance has the following benefits:-

- Mappings are possible where the addressed entities change over time
- Late binding of a reference to its interpretation, e.g. via content negotiation.
- The author may refer to the concept, rather than to the singular representation of the concept. Hence, when the representation changes, there is no need to change the resource.

The resource identifier identifies the particular resource that is involved in the communication between components. The naming authority of the resource identifier needs to maintain the semantic validity of the mapping over time to ensure that the correct entity is addressed, always.

Representations form means to capture resources and to transfer them between components, e.g. files, HTTP message entities, etc. Representations usually consist of data and metadata that describes that data. The data format of a representation is its media type, e.g. text/html .

REST Connectors implement abstract interfaces for component communication. Fielding (2000) defines the following connector types:

Connector	Modern Web Examples
client	libwww, libwww-perl
server	libwww, Apache API, NSAPI
cache[11]	browser cache, Akamai cache network
resolver[12]	bind (DNS lookup library)
tunnel[13]	SOCKS, SSL after HTTP CONNECT

Figure 12: REST connector types.

These connectors carry out special network communication tasks for a REST component. As all REST communication is stateless, each request contains all information that a connector needs to understand the request, independent of any earlier requests. Hence:-

- There is no need for the connectors to retain application state between requests.
- Interactions may be carried out in parallel.
- In intermediary, e.g. for rearranging requests dynamically, may do this because he understands a request in isolation.
- The response may be cacheable.

[11] Cache connectors may be located on client or on server side to save cacheable responses for later reuse.

[12] Resolvers translate partial or complete resource identifiers into network address information.

[13] Tunnels relay communication across a connection boundary; e.g. providing encrypted communication between two enterprise sites.

Connector interfaces need to provide interfaces for the following input types:-

- Request control data.
- Resource identifier that indicates that target of the request.
- An optional representation.

A connector's output needs to contain the following output types:-

- Response control data.
- Optional resource metadata.
- An optional representation.

If necessary, request processing can be invoked before the request is completely known, as the connector invocation is synchronous.

Client and server form primary connector types, whereas the client initiates a communication by making a request whereas a server listens and responds to received requests in order to supply access to its services. A REST component may include both client and server connectors.

REST components form special roles in the overall application action. There are the following REST components:

Component	Modern Web Examples
origin server	Apache httpd, Microsoft IIS
gateway[14]	Squid, CGI, Reverse Proxy
proxy[15]	CERN Proxy, Netscape Proxy, Gauntlet
user agent	Internet Explorer, Chrome, Firefox

Figure 13: REST components.

7.2 Architectural Views

Architectural views describe how elements act together to form an architecture. Fielding states that the following views are useful for modelling REST application architectures:-

- **Process View**
  A process view elicits primarily the interaction relationships between components and the data flows through the system. Typical elements of process views are e.g. the REST components shown above.
- **Connector View**
  These views concentrate on the communication mechanics and procedures between components.

[14] A gateway provides an interface encapsulation to other services for e.g. data transformation or performance enhancement or security enforcement. A gateway is imposed by the network.

[15] A proxy is a gateway that is voluntarily selected by the client.

- **Data View**
  Data views concentrate on the application state; the design principle of statelessness requires that there is a default "ready" state when there are no outstanding requests.

These concepts prepare the use of REST for the modelling platform.

7.3 Model checking requirements

In MSEE, the Modelling tool is proposing interface for creating, editing and storing a model at BSM and TIM level (Modelling step in Figure 10 upper part).

But also the need of model verification and validation in the Modelling platform has been identified. Different kind of validation can be used. First syntactical verification can be provided by verifying model properties regarding constraints of the formalism. Nevertheless this approach doses verify dynamic inconsistency like exception that can occur in java programming. In the objective of getting some dynamical validation, the proposition is to test the model in simulation. For the behavioural point view the descriptions at TIM where the formalism is BPMN level are not sufficient to describe the temporal causalities, it needs to be completed by behavioural descriptions.

Several formalisms coexist to describe the behaviour of the models. For instance, IDEF3 state model, State Chart diagrams in UML, State machine. We propose to focus on DEVS that proposes to describe explicitly the notion related to time and define time life duration associated to the model state. We will propose in the following a short recall of this formalism. In addition the communication for the model checking should be independent of the model and eventually modified according the desired behaviour to obtain. The proposition is to validate the model uncorrelated to a checking facility. We propose the use of REST to communicate with a server that will host the validation device. REST will propose to charge the local state and scenario to test (i.e. Figure 14 lower part).

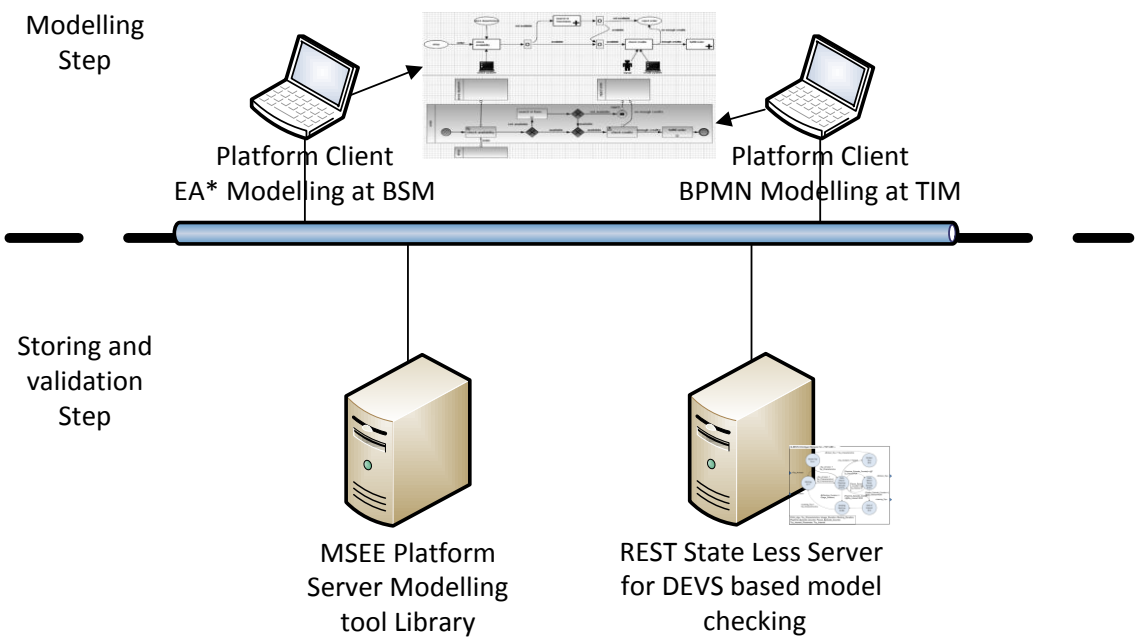

Figure 14: Servers calls and answers

7.3.1 Existing Service modelling techniques

Service Modelling is a recent domain it has not adopted yet a unique common standard for developing frameworks to execute services processes. There is a lack of one sufficient Modelling language for Service or either Product and Service Systems PSS (Becker 2009). The specification of a Service Modelling may involve different process, application and actors components essential to its execution but heterogeneous. The specification standards are numerous; (Becker 2009) reports more than 15 main reference models. To start on Modelling service some authors (e.g. (Meier 2010), have reused by imitation the process sequence of workflow with the use of service modules. Moreover, (Alix 2012) have chosen in a previous work the graphical definition of the Service-Oriented Modelling Framework (SOMF) (Bell 2008). The reasons were the coverage of the domain and the user-friendly design. In any case, the problem is the correctness checking or the scenario evaluation on the service models. Concretely, this field is missing verification and validation methods; the simulation can give predictive information on the correctness of the models.

The W3C also proposed a XML representation of Service Modelling Language (SML) (W3C 2007) that is accepted as a standard in Service Modelling community. The XML Service Modelling process model structure correctness can be certified by referring to a Service Modelling Document Type Definition (DTD). However, this XML representation is not fully convenient for the XML specification of a PSS Model. The description is more oriented to Computer Science Service Modelling than industrial PSS oriented. For instance, the Service participants (users) are not taken into account.

7.3.2 Existing Service Simulation frameworks

The literature reports on existing projects that attend to model and simulate Product and Service. In particular the study of the dynamic behaviour of the PSS can provide information to the designers on how to handle the system and to verify properties (Sakao 2009). In (Phumbua, 2010), the authors give a state of the art on Discrete-Event System Modelling of PSS. They recalled one approach that studies PSS scenarios with accidental events. Two other approaches are studied; they are focused on the lifecycle of the PSS in order to follow indicators during design and, production (Fujimoto 2003) and (Komoto 2003). In (Meier 2010), the behaviour is pointed with the implementation of agents. The different research identify the variables to follow during simulation including the price, process costs lifetime, sales frequency, lifetime, etc. In the following, the focus is done on the simulation of the price of the product and on the time of use of the product in order to compute its impact on the environment.

7.3.3 PSS G-DEVS/HLA Modelling & Distributed Simulation

The simulation requires interoperating and synchronizing with heterogeneous and distributed components/actors that each implements a subset of the PSS. In addition, actual complex industrial service processes are composed of heterogeneous and distributed resources (material, immaterial and human in the loop). To address these requirements of interoperability, research works introduced a Service Modelling Environment G-DEVS/HLA compliant (Alix 2012) able to interoperate simulation components, human–computer interactions or tactile sensors. This section presents a PSS Modelling Checker Specification language and on Distributed G-DEVS Simulation.

7.3.4 DEVS and G-DEVS

B.P. Zeigler (Zeigler 2000) defined in early 70s a formal specification of real systems with discrete event entitled Discrete Event System Specification (DEVS). DEVS is defined as an abstract universal formalism that is independent of implementation. An atomic DEVS is a block model with input and output ports to exchange discrete events. The behaviour of the model is described by states linked by transitions. The transition is fired by receiving discrete events, which causes the current state change to a next one. In addition to major discrete event Modelling techniques, DEVS presents the possibility of autonomous evolution of the model through the state time life; associated with the life extinction of the current state an internal state transition function is triggered. The real world input, output signals and states are abstracted by piecewise constant values where thresholds are considered as discrete events. The concept of coupled models, introduced later, provides a means to build new coupled models made by reusing and connecting stored models. The DEVS simulator is explicitly specified in the formalism and permit to obtain unambiguous simulator development.

The Generalized Discrete Event Specification (G-DEVS) formalism in traduced by Giambiasi in the 90's (Zacharewicz 2008) is chosen for its formal properties to describe systems with discrete event models and its capacity of simulation. This formalism emerged with the drawback that most classical discrete event abstraction models (e.g. DEVS) face: the majority approximate observed input–output signals as piecewise constant trajectories. G-DEVS defines abstractions of signals with piecewise polynomial trajectories. Thus, G-DEVS defines the coefficient-event as a list of values representing the polynomial coefficients that approximate the input–output trajectory. G-DEVS is a general specification language that clearly separates Modelling and simulation processes. In detail, G-DEVS privileges the use of two concepts: event and timed state. The debate is about the explicit use of this formalism in the tool. Or should it be automatically generated from the BPMN

7.3.5 DEVS automatically generated from the BPMN

Some works has already been proposed to almost automatically generate DEVS from the BPMN Workflow (Zacharewicz 2008).

Workflow Management Coalition (WfMC) provides a good framework to develop business process. The description of a Workflow may involve a process model, different programs, and actors which are essential to its execution. This description is user-oriented and does not need to develop programming code (it can be automatically generated from a graphical description). But the drawback is there is no clear simulation semantics associated to these Workflow engines. Almost of these engines are ad hoc. This fact may lead to errors that are difficult to detect.

DEVS, Statecharts, Petri nets are well-known formalisms to describe the behaviour of complex discrete event systems. They give formal frameworks in which modelling and simulation processes are clearly separated. DEVS seems to be more general and flexible than the other formalisms. However Workflow users are not familiarized with DEVS. Thus we propose a set of rules (grouped in form of an algorithm) that transforms automatically a Workflow specification into a G-DEVS model. Results have been published in (Zacharewicz 2008).

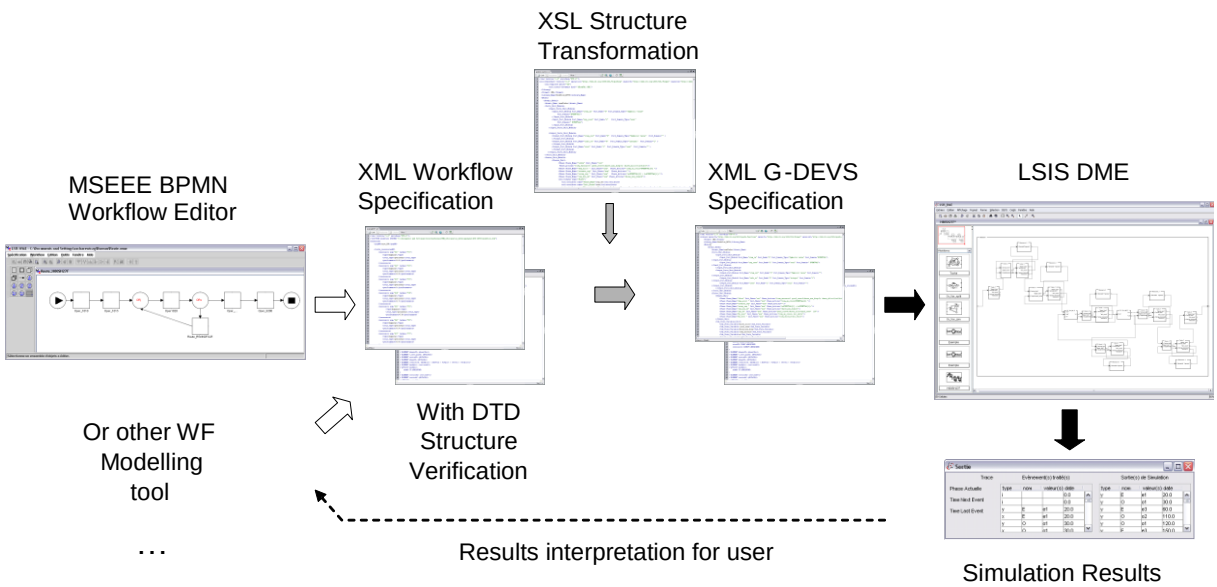

Figure 15: From BPMN Workflow to DEVS models

7.3.6 Distributed Simulation with HLA

The Service system is complex. It does not appear realistic to try to completely tackle by one model. The idea is to combine behavioural deterministic models (here G-DEVS) with BPMN. Based on the experience of distributed simulation, it can be proposed to address the interoperability of the components by conforming to the distributed simulation standard High Level Architecture (HLA) to support the REST architecture. HLA is a software architecture specification that defines a common understanding to create a global simulation composed of distributed simulations (or other software components). In HLA, every participating simulation is called federate. A federate interacts with other federates within an HLA federation, which is in fact a group of distributed federates. The HLA set of definitions brought about the creation of Standard 1.3 in 1996, which then evolved into HLA 1516 in 2000 (IEEE 2010) and HLA 1516 Evolved in 2010 (IEEE.1 2010). Finally, we define a distributed Service Modelling Environment that interfaces components for Modelling and validation in HLA compliant Federation.

7.4 REST in action for Model checking and validation

REST can support taking into account the local behaviour of a service client in the relation with the server. The REST-style architectures that consist of clients and servers can be applied to the architecture of the modelling platform in MSEE. The idea is to control the access of the modelling resources to the validation of the models trough some REST calls. These kinds of calls can be seen as light communication that keeps the communication flow flowing. The different levels of Modelling can give solicitation to validations facilities hosted by the server. It means also that validation process can be adapted on the server validation tool.

The application can control the models consistency that can be stored locally thanks to a call to the platform server to get some properties validation. No state of the calls will be stored on the platform only locally. For the validation the models can be confronted to reference models that can be structured thanks to ontology relations and rules.

7.4.1 Checking structure

Modelling clients will be proposed under a web browser shape or a dedicated application (refer to D15.2) that will call remote resources and initiate requests to servers. Servers process requests and return appropriate responses. For instance the calls can be opening model library or storing model that is a classical access to remote resource. The modelling aspects and the transformation of the models have been introduced in the deliverables. In this deliverable we discuss the requests and responses that are built around the transfer of dynamical representations of the model resources. A resource can be essentially any coherent and meaningful concept that may be addressed regarding its consistency with the perspective of the temporal aspect. A representation of a resource is typically a document that captures the current or intended state of a resource. At any particular time, a client can either be in transition between application states or "at rest". A client in a rest state is able to interact with its user, but creates no load and consumes no per-client storage on the set of servers or on the network.

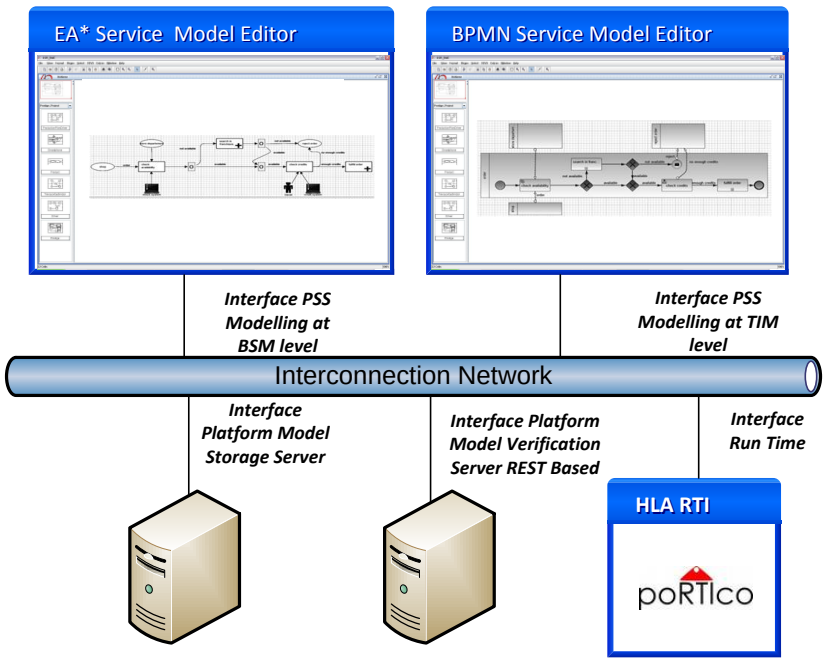

Figure 16: Servers Technical Architecture calls and answers

The REST based platform can receive calls from users in the idea of getting behavioural validation. The client will select the part of the model he would like to obtain dynamical validation. It transmits its current state and time information's and events that describe the scenario required for validation.

REST gives the content of the communication. It misses the way to exchange the messages. For instance in distributed simulation a long experience has been developed for handling message exchanged between distributed entities. For Example the HLA provide a standard in this domain. It helps also for the messages synchronisation that can be helpful for the organizing the concurrent call to a resource. The central component in HLA is the Run Time infrastructure (RTI). In the Figure 16 the open Source pRTIco software is suggested to achieve that goal.

7.4.2 G-DEVS Validation

As an answer to that call, the server will deliver some simulation results useful to validate the model. REST can be seen as a dedicated communication link with a model checking server. This server can host DEVS simulator (Zeigler 2000) for model validation of desired properties with simulation. Several works illustrate the implementation of these techniques. This is making a link with the WP12 to make the link with the service engineering and simulation. The server takes into account the local model (it can be seen as a local state). Then it will trigger the model checker in the server dedicated.

The figure 13 is illustrating this process. In particular, it describes the server side that is hosting the model checker and the simulator to test properties of the local model. It stresses that due to the REST architecture it does memorize the actual state of the local models and the event list to describe the scenario. This principle is reducing the charge of the server.

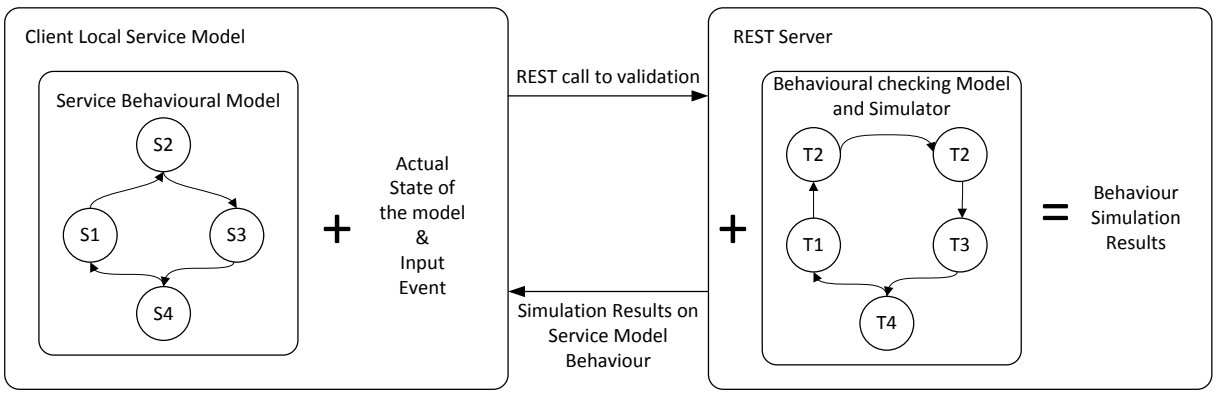

Figure 17: Servers Model Checking calls and answers

This section has given some specification on the possible implementation of the behavioural model checking using G-DEVS. This feature can be proposed for the V2 of the Modelling platform.

8 Platform Architecture Proposed Features

This section shows the different specification in the context of their use and illustrate with their environment of use. The Figure 18 is describing a synthesis big picture of what is proposed to the platform in term of interfaces.

First the ontology with the Mentor approach is proposed to creating reference ontology and to facilitate the model transformation thanks to semantic approach. The step 0 is proposing to reference the models according to reference ontology it uses the MENTOR approach to match ontologies and to extract a Meta ontology. The step 1 is refereeing to the MO ontology that enables to relate different models and retain traceability information (can also relate to the reference ontologies).

The second dimension discussed is the matching of BSM and TIM formalisms matching with USDL. The idea is to provide gateway to service domain at business level for instance using one of the reference in the domain: USDL. The proposition is to adapt EA* and BPMN concepts to match the USDL description. Nevertheless this work has identified some mismatch between these description languages. It means that the model formalisms need to be enriched by other description variables for instance in the finance and legal domain. The figure is describing mostly unidirectional arrows to offer service but it can be also admitted to receive data in USDL format that could integrated to models in the MSEE modelling platform.

Thirdly REST communication is proposed to support the models verifications and validation regarding states and temporal considerations thanks to simulation. In detail, at the bottom of TIM level a two directions path can be proposed. The first way is the one that goes to implementation of service; it is discussed in the 15.2 deliverable and not detailed in this one. The second way is a model transformation to behavioural models that can be an issue to check and validate the models based on the dynamic of their use. This verification is useful to test the models and to anticipate undesired behaviour before their implementation.

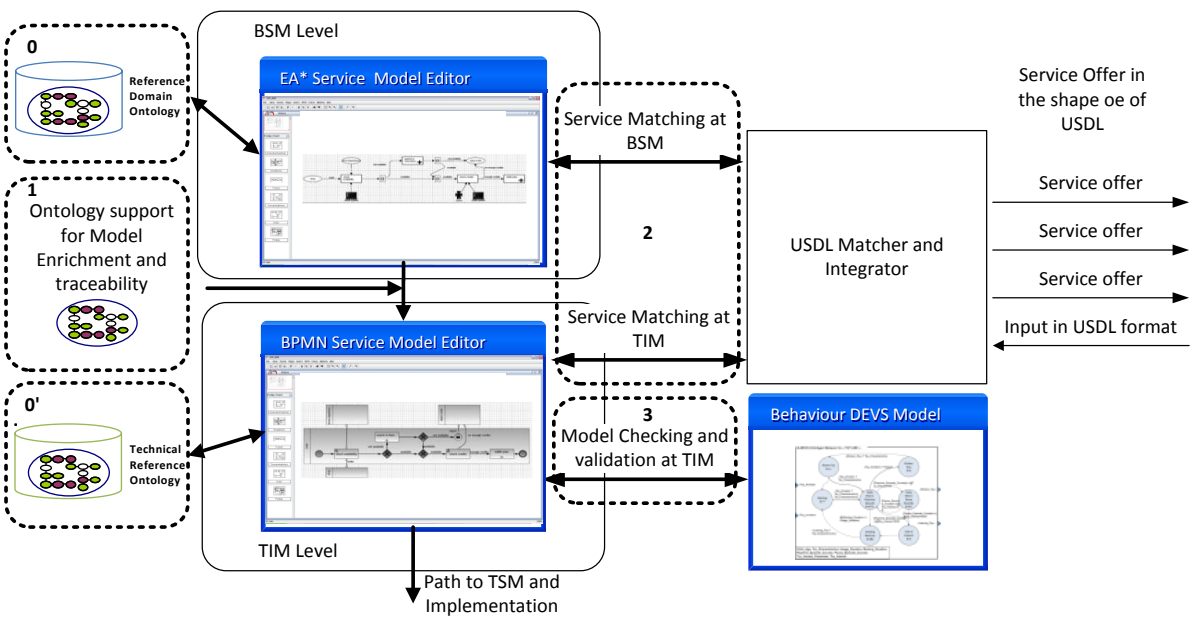

Figure 18: D11.5 Big picture

9 Conclusion

This deliverable has permitted to make explicit some aspects of the modelling platform that were not tackled in previous specification documents of the project.

In particular it explicitly describes the link with USDL and ontology. Also it describes architecture for model checking beads on REST and using the DEVS formalism to describe models behaviour and checking.

10 References

Agostinho, C., Correia, F. & Jardim-Goncalves, R. (2010) Interoperability of Complex Business Networks by Language Independent Information Models. In 17th ISPE International Conference on Concurrent Engineering (CE2010). Cracow, Poland: Springer-Verlag. Available at: http://www.springerlink.com/content/q775010855762526/.

Alix T., G. Zacharewicz: Product-service systems scenarios simulation based on G-DEVS/HLA: Generalized discrete event specification/high level architecture. Computers in Industry 63(4): 370-378 (2012)

Alter, S., Service system fundamentals: Work system, value chain, and life cycle, IBM Systems Journal, 2008. 47(1): p. 71-85.

Becker J., D.F. Benerungen, R. Knackstedt, The challenge of conceptual modeling for product-service systems: status-quo and perspectives for reference models and modeling languages, Information Systems and E-Business Management Journal, 8 (1) (2009), pp. 33–66

Bell M., Service-Oriented Modeling: Service Analysis, Design, and Architecture, John Wiley & Sons, Inc., Hoboken, NJ (2008).

Boudjlida.N and al. 2007, Experimental Semantic Enrichment of Enterprise Models for Interoperability and its Practical Impact. INTEROP Deliverable DTG4.2

Donini, F. M.; Lenzerini, M.; Nardi, D.; Schaerf, A. (1991). A hybrid system with datalog and concept languages: Trends in Artificial Intelligence, Volume LNAI 549, Springer Verlag

Doumeingts G., Ducq Y. (2001) Enterprise Modelling techniques to improve efficiency of enterprises. in International Journal of Production Planning and Control - Taylor & Francis – Vol. 12, number 2, Mars 2001, pp 146-163

Ferrario R., N. Guarino, "Commitment-Based Modeling of Service Systems", in M. Snene (ed.), IESS 2012, International Conference on Exploring Services Science , Springer Verlag, Lecture Notes in Business Information Processing, vol. 103, Berlin Heidelberg 2012, pp. 170-185.

Fielding, R.T. (2000): Architectural Styles and the Design of Network-based Software Architectures. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (08.10.2012).

J. Fujimoto, Y. Umeda, T. Tamura, T. Tomiyama, F. Kimura, Development of service oriented products based on the inverse manufacturing concept, Environmental Sciences & Technology, 37 (2003), pp. 5398–5406

Grangel, R., Correa, K., Antonio, F.D., Bourey, J.-pierre & Berre, A.J. (2007) Analysing CIM2PIM Approaches to Improve Interoperability. In 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA'07). Funchal, Madeira.

Grangel, R., Salem, R., Bourey, J.-pierre, Daclin, N. & Ducq, Y. (2007) Transforming GRAI Extended Actigrams into UML Activity Diagrams: a First Step to Model Driven Interoperability. In 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA'07). Funchal, Madeira.

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specifications. Journal of Knowledge Acquisition, vol.5(2), pp. 199-220

Hayek, F. (1945). "The Use of Knowledge in Society". In: The American Economic Review, Volume 35, Issue 4 (Sep., 1945), 519-530

IEEE std 1516.1-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification, Institute of Electrical and Electronic Engineers, New York, NY, 2010.

IEEE std 1516.1-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture

(HLA) – Federate Interface Specification, Institute of Electrical and Electronic Engineers, New York, NY, 2010

ITU (2006): X.711 : Information technology - Open Systems Interconnection - Common Management Information Protocol: Specification. http://www.itu.int/rec/T-REC-X.711/en/ (last update: 03.10.2006).

Jouault, F. & Kurtev, I. (2007) On the interoperability of model-to-model transformation languages. Science of Computer Programming, 68, p.pp.114-137.

Komoto H., T. Tomiyama, M. Nagel, S. Silvester, H. Brezet, Life cycle simulation for analyzing product service systems, The Fourth International Symposium on Environmental Conscious Design and Inverse Manufacturing (2005)

Meier H., R. Roy, G. Seliger, Industrial product-service systems-IPS2, CIRP Annals – Manufacturing Technology, 59 (2) (2010), pp. 607–627

Mosterman, P.J. & Vangheluwe, H. (2004) Computer Automated Multi-Paradigm Modelling: An Introduction. SIMULATION, 80 (9), p.pp.433-450.

OMG (2011a) Business Process Model and Notation (BPMN) - version 2.0 (formal/2011-01-03). Available at: http://www.omg.org/spec/BPMN/2.0/PDF/.

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 (2008) Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification - Version 1.0 (formal/2008-04-03). Available at: http://www.omg.org/spec/QVT/1.0/PDF/.

OMG (2011b) OMG Unified Modelling LanguageTM (OMG UML), Infrastructure - version 2.4.1. Available at: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/.

OMG (2010) Object Constraint Language - version 2.2 (formal/2010-02-01). Available at: http://www.omg.org/spec/OCL/2.2/PDF/.$

Phumbua S., B. Tjahjono, Simulation modelling of product-services systems: the missing Link, MATADOR Conference (2010)

Sakaoa T., Y. Shimomurab, E. Sundin, M. Comstock, Modeling design objects in CAD system for Service/Product Engineering, Computer-Aided Design, 41 (3) (2009), pp. 197–213

Salem, R., Grangel, R. & Bourey, J.-pierre (2008) A comparison of model transformation tools: Application for Transforming GRAI Extended Actigrams into UML Activity Diagrams. Computers in Industry, 57 (7), p.pp.682-693.

Sarraipa, J., Jardim-Goncalves, R. and Steiger-Garcao, A. (2010). MENTOR: an enabler for interoperable intelligent systems. International Journal of General Systems, Volume 39, Issue 5 July 2010, pp 557 –573

Sarraipa J., R. Jardim-Gonçalves, T. Gaspar, and A. Steiger-Garção, "Collaborative ontology building using qualitative information collection methods", in Proc. IEEE Conf. of Intelligent Systems, 2010 pp.61-66

Thomas, D. & Barry, B.M. (2003) Model Driven Development: The Case for Domain Oriented Programming. In Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA'03). ACM Press.

USDL Website, http://www.internet-of-services.com/ (accessed 12/11/2012)

USDL 3.0 M5 Specification Archive**,** http://www.internet-of-services.com/fileadmin/IOS/user_upload/pdf/USDL-3.0-M5-Archive.zip

W3C, SOAP, SOAP specifications. http://www.w3.org/TR/soap/ (accessed 13.11.2012).

W3C, SML, SML specifications, http://www.w3.org/TR/sml/ (accessed 13.11.2012).

W3C, USDL, Incubator discussion, http://www.w3.org/2005/Incubator/usdl/ (accessed 13.11.2012).

Zacharewicz G., C. Frydman, N. Giambiasi, G-DEVS/HLA environment for distributed simulations of workflows, Transactions of the SCS, Simulation, 84 (5) (2008), pp. 197–213

Zacharewicz, G., Chen, D. & Vallespir, B. (2009) Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability. In I-ESA'09. Benjing, China.

B. Zeigler, T.G. Kim, H. Praehofer, Theory of Modeling and Simulation, (second ed.)Academic Press, New York (2000) ISBN 978-0127784557