SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

EXTENDING AND UTILIZING THE SOFTWARE AND SYSTEMS PROCESS ENGINEERING METAMODEL WITH ONTOLOGY

DISSERTATION THESIS

ID: FIIT-3094-4984

Study programme: Software Engineering
Research field: 25-31-9 Program and Information Systems
Department: Institute of Informatics and Software Engineering
Supervisor: Professor Dr. Pavol Návrat

Bratislava 2010 **Ing. Miroslav Líška**

Anotácia

Modelom riadená architektúra (MDA) poskytuje množstvo známych metamodelových štandardov ako MOF, UML, BPMN, SPEM a iné. Vo všeobecnosti, metamodel definuje akým spôsobom je možné použiť model určitého jazyka. Bohužiaľ, uvedené metamodely majú semiformálnu architektúru, a teda nie je možné overovať vytvorené jazykové konštrukcie pomocou formálnych metód ako napr. verifikácia konzistentnosti, či splniteľnosti. Týmto problémom sa zaoberá množstvo výskumných prác, ktoré navrhujú použitie rôznych formálnych metód. V poslednej dobe sa hlavným smerom výskumu stáva kombinácia MDA a Sémantického webu, kde sa kladie dôraz na spracovávanie údajov z ohľadom na ich význam. No pravdou je, že väčšina prác sa zaoberá hlavne jazykmi UML a OWL, keďže možnosti obidvoch týchto jazykov sú z pohľadu špecifikácie nejakej domény podobné.

Predmetom tejto dizertačnej práce je prezentovať našu metódu transformácie SPEM metamodelu do technického priestoru Sémantického webu a jeho využitia vo vybraných disciplínach softvérového inžinierstva. SPEM je jazyk určený na definovanie softvérových a systémových procesov a ich komponentov. Posledná verzia SPEM 2.0 prezentuje jeho nové kľúčové vlastnosti, ako napr. úplné oddelenie definície metódy od jej aplikácie v procese a podobne. Keďže sme zatiaľ nenašli žiadnu prácu, ktorá by prezentovala obsiahlu analýzu SPEM metamodelu pre potreby jeho transformácie do Sémantického webu, a ani sme nenašli žiadnu SPEM ontológiu, ktorá by zohľadňovala SPEM metamodel s jeho možnosťami, rozhodli sme sa zaplniť tento chýbajúci priestor našim návrhom. Najskôr prezentujeme vytvorenie SPEM ontológie a následne navrhujeme jej priame použitie pre: validáciu SPEM modelov v ontológii, ontológiou orientovanú metódu projektového plánovania a metódu pre zladenie softvérového projektu s dodávateľom. Práca je doplnená o príklady v deskriptívnej logike spolu s implementáciou, ktorá prezentuje konkrétne použitie navrhovaných metód.

Kľúčové slová: SPEM, OWL, Metamodel softvérového a procesného inžinierstva, Jazyk pre webové ontológie, validácia modelu, verifikácia projektového plánu, zladenie softvérového projektu

Annotation

Model Driven Architecture (MDA) provides a set of metamodel standards as MOF, UML, BPMN, SPEM and others. A metamodel in general makes statements about what can be expressed in the valid models of a certain modeling language. Unfortunately, the mentioned metamodels have semiformal architecture, thus it is not possible to make and to verify created language statements with formal techniques such as the consistency or satisfiability verification. Many research works have proposed solutions that address to this problem with wide range of created formal techniques. Recently, the combination of MDA and the Semantic Web, in which data processing is concerned with regard to their semantics, become the leading subject in this direction. However, the works mostly focus only on UML-OWL relationships, since the possibilities of these languages are similar in context of domain specification.

In this thesis, we present an approach of SPEM transformation to the Semantic Web technical space and its utilizations. SPEM is used to define software and systems development processes and their components. The latest version of SPEM 2.0 has defined several new key capabilities such as clear separation of method content definitions from the development process. Since we neither have found a work that presents a comprehensive analysis of the SPEM metamodel in order to transform it to the Semantic Web technical space, nor a SPEM ontology that is conformed to the SPEM metamodel with respect of the SPEM capabilities, we have decided to bridge this gap with our proposal. First we present the creation of a SPEM ontology and then we present its three utilizations that are: a SPEM model validation with ontology, an ontology based approach to project planning and an approach to software project enactment with a supplier. The thesis is supplemented with examples in a description logic with implementation that illustrates the concrete use of proposed methods.

Keywords: SPEM, OWL, Software and Systems Process Engineering Metamodel, Web Ontology Language, model validation, project plan verification, software project enactment

Acknowledgement

First, I would like to thank my supervisor Professor, Dr. Pavol Návrat. Your support, enthusiasm and imagination were the most valuable help as I ever got received in my studies. Since I am conscious that it was only fraction what I caught to learn from you, I pray that a future will bring me an additional opportunity.

I would also like to thank my chiefs in Datalan, especially: Norbert Lacko, Michal Klačan, Luboš Petrík and Milan Lešták that have allowed me to study, even my activities were not always in day offs. I appreciate it very much and in exchange, I am ready to pay my dept. My mind and heart are full of ontologies. Now, they are yours.

Finally, none of this would be possible without my wife Katarína. You have always been there for me and I cannot imagine this degree or this life without you.

Dedication

to my beloved wife Katarína
and son Slavomír

List of Figures

List of Tables

List of Abbreviations

SWEBOK	Software Engineering Body of Knowledge
SPEM	Software and Systems Process Engineering Metamodel
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
MDA	Model Driven Architecture
MOF	Meta Object Facility
UML	Unified Modeling Language
RUP	Rational Unified Process
EMF	Eclipse Modeling Framework
XML	Extensible Markup Language
XMI	XML Metadata Interchange
XSLT	Extensible Stylesheet Language Transformations
DL	Description Logic

Contents

Part I
The Problem

1 Introduction

The software engineering is application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (IEEE, 2004). Despite the fact that at present there exist many software developments process frameworks, the fact that relatively few projects are completely successful is an indicator of the difficulty of the task (Kruchten, 2003). One of the problems is that standard software development process frameworks are usually used as a navigable websites that contain only human-readable descriptions with supporting materials as documents templates etc. Thus, these kinds of frameworks cannot be used to represent machine interpretable content (Fujita, 2008). Moreover, these process frameworks are used in the technical spaces (Kurtev, 2002) that have model based architecture, such as MDA or EMF (Steinberg, 2009). These kinds of technical spaces also limit knowledge based processing, owing to their weakly defined semantics (Gašević, 2009). Moreover the difficulty of software development is greatly enhanced when it is inevitable to cooperate with a supplier. The general issue is to manage a lot of differences such as different tasks, software work products, guidelines, roles etc (IEEE, 2004).

1.1 Motivation

However, at present the emerging field of Semantic Web technologies promises new stimulus for Software Engineering research (Happel, 2006). The Semantic Web is a vision for the future of the Web, in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web (McGuiness, 2004). The today’s key Semantic Web technology is OWL. It is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans (Smith, 2004).

Thus if we transform a definition of a software method to the Semantic Web technical space, we can use many knowledge oriented techniques to maintain them. In this work we address such opportunity. We use SPEM, the MDA standard used to define software and systems development processes and their components (OMG, 2008). We transform SPEM from the MDA technical space to the Semantic Web technical space; so we can work with SPEM as with an ontology. Based on this transformation, we propose three approaches that utilize created SPEM ontology in the context of selected SWEBOK knowledge areas that are: model validation, project verification and software project enactment with a supplier (IEEE, 2004).

Likewise as another MDA' standard UML have brought a new generation into specification, documentation and realization of information systems, it is very probable that SPEM will play the same role in the domain of software process engineering. Thus we can suppose that a SPEM Ontology has at least the same potential, since it improves the SPEM metamodel, therefore its conceptual framework with ontology either.

1.2 State of the art and related works

The thesis presents an approach to extending and utilizing the Software and Systems Process Engineering Metamodel 2.0 with OWL-DL ontology. The subject is a SPEM Ontology development from the SPEM metamodel, thus we present the definitions of the terms *metamodel*, *model* and *ontology* first. Since the thesis subject is usability of ontologies in the software engineering domain, second we provide a survey of published works that covers this area. The last part is focused on the works that combine MDA and the Semantic Web where this thesis belongs to.

1.2.1 Metamodel vs. Model vs. Ontology

A general definition of metamodels is given in the OMG's MOF specification: A ***metamodel*** is a model that defines the language for expressing a model (OMG, 2006a). Another definition states that metamodels represent and specify models; that is, they describe about what are the valid ingredients of a model (Callero, 2006). Additional definition states that a metamodel makes statements about what can be expressed in the valid models of a certain modeling language. Hence, a meta-model is a prescriptive model of a modeling language

(Seidewitz, 2003), thus the metamodel prescribes the structure or behavior of reality and reality is constructed according to a model (Callero, 2006).

The second notion ***a model*** is a representation of reality intended for some definite purpose (Pidd, 2000). Other author's definition states that a model is an external and explicit representation of a part of reality as seen by the people who wish to use that model to understand, change, manage, and control that part of reality (Pidd, 2000).

The last notion ***an ontology*** is formal explicit specifications of a shared conceptualization (Gruber, 1993). Other definition states that ontology is a shared, descriptive, structural model, representing reality by a set of concepts, their interrelations, and constraints under the open-world assumption (Callero, 2006). Likewise, another definition states that ontology can be seen as the study of the organization and the nature of the world independently of the form of our knowledge about it (Guarino, 1995). The final definition mentioned in this these states that ontology is a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic (Hendler, 2001). Hence, an ontology is descriptive model of reality, but reality is not constructed from it (Callero, 2006).

Even the previous definitions of the same concept are slightly different we use them as consistent. However, a legitimate question is raised, why to transform a metamodel to an OWL-DL ontology? Which contributions can such transformation provide? We discuss the answers in the subchapter 2.1, which explains the motivation for the SPEM metamodel transformation to an OWL-DL ontology.

1.2.2 Ontologies in software engineering

Usability of ontologies in software engineering and technology (SET) can be distinguished into two main categories: SET domain ontologies and ontologies as software artifacts (Callero, 2006). SET domain ontologies refer to the ontologies whose main goal is to represent (at least partially) knowledge of a certain subdomain within SET matter. In the second category, ontologies are used as software artifacts of diverse types, in some software process. Since this thesis concerns with the former category, we do not deal with the latter. Based on the SWEBOK guide, prototypes of ontologies for the representation of the complete software engineering domain have been created (Mendes, 2005), (Sicilia, 2005). Other ontology that also conceptualizes the software engineering domain, is OntoGLOSE (Hilera, 2005), created and based on (IEEE, 2002). Falbo et al. (1992) and Larburu et al. (2003) have proposed ontologies to model the knowledge related to the software process, including concepts such as Life Cycle Model, Software Process, Activity, Procedure, Task, Role, or Artifact, among others.

1.2.3 MDA and the Semantic Web

Since we want to use SPEM in the technical space of the Semantic Web and SPEM is MDA based, we can utilize research results on using MDA in the technical space of the Semantic Web. SPEM is specified in MOF language (OMG, 2006a) that is the key language of MDA. MOF is a language for metamodel specification and it is used for specification of all model-based MDA standards (Frankel, 2003). It provides metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems. On the Semantic Web side, OWL is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. OWL is based on RDFS (Brickey, 2004). Both MOF and RDFS provide language elements, which can be used for metamodeling. Although they have similar language concepts such as `mof:ModelElement` with `rdf:Resource`, or `mof:Class` with `rdf:Class`, these languages are not equivalent. RDFS, as a schema layer language, has a non-standard and non-fixed-layer metamodeling architecture, which makes some elements in model to have dual roles in the RDFS specification (Pan, 2001). MOF is also used for specification of UML, a language for specification, realization and documentation of software systems (OMG, 2009a). Even if UML and RDFS are similar in a domain of system specification, they are also substantially different. One issue that has been addressed was the problem that RDF properties are first class entities and they are not defined relative to a class. Therefore a given property cannot be defined to have a particular range when applied to objects of one class and another range when applied to objects of a different class (Cranefield, 2001). Note this difference have also been propagated between OWL and UML (Hart, 2004). At present the main bridge that connects the Semantic Web with MDA is stated in the ODM (OMG, 2009b). ODM defines the OWL Meta-Model specified in MOF (MOF – OWL mapping) and also the UML Profile for Ontology modeling (UML – OWL mapping). This architecture can be extended with additional mapping between the UML Profile for OWL and other UML Profile for custom domain (Gašević, 2009); hence the works provides the major bridge between OWL and another MDA specification.

1.2.3.1 SPEM and the Semantic Web

This chapter presents the works that are the most closest to this thesis, because they also propose SPEM utilizations in the Semantic Web technical space. The *first* work proposes to represent SPEM in DL (Wang, 2006). The work creates mapping from MOF to DL and mapping from OCL (OMG, 2006b) constraints of SPEM to DL. The reason for the former mapping is to represent the SPEM MOF based metamodel with DL and the latter is to represent additional OCL constraints that supplement the SPEM metamodel with additional semantics. The result of these mappings is a formal metamodel of SPEM in DL that is presented in Figure 1. The work additionally presents a simple example of reasoning with the proposed SPEM DL representation. The example covers an information system delivery process. The results of reasoning are discussed, for example that the inconsistency exists when a phase "Preliminary Analysis" does not have assigned a performer.

$$\begin{aligned}
&\text{WorkDefinition} = \text{Operation}\\
&\qquad \sqcap\ \exists\ \text{parentWork.WorkDefinition}\\
&\qquad \sqcap\ 1\ \text{performer.ProcessPerformer}\\
&\qquad \sqcap\ \exists\ \text{subWork.WorkDefinition}\\
&\text{Activity} = \text{WorkDefinition}\\
&\qquad \sqcap\ \exists\ \text{assistant.ProcessRole}\\
&\qquad \sqcap\ 1\ \text{step.Step}\\
&\text{Step} = \text{ActionState}\ \sqcap\ 1\ \text{stepActivity.Activity}
\end{aligned}$$

Figure 1. Excerpt of the SPEM metamodel represented in DL

The *second* work proposes SPEM process constraint definition with the semantic rules with SWRL (Rodríguez, 2009). Note that SWRL is W3C Semantic Web Rule Language that combines OWL and RuleML (Horrocks, 2004). The work introduces that the most SPEM terms can be directly translated into OWL. Additionally it states that the translation does not aim at substituting the original model, because it serves as a complement for adding reasoning and inference support to SPEM based models. A prototype design of an engine for process constraint verification is presented. The main idea is to verify consistency between SWRL process constraints that can be obtained from the EPF Composer and SWRL process constraints from a project plan. By other words the work intends to verify a project plan with a SPEM process using SWRL. Note that the EPF composer enables to serialize SPEM models with XMI, hence to the XML representation of an ontology language either (Kurtev, 2003).

The *third* work also intends to use SPEM in the Semantic Web technical space. It presents a competency framework for software process understanding (Zualkernan, 2008). The motive is to create assessments for a correct understanding of a process that can be used in a software development company. The paper introduces creation of SPEM software process ontology for the SCRUM (Schvaber, 2002) software process with EPF Composer again. Consequently paper presents generation of assessments that were generated with the IMS QTI, and XML based e-Learning standard with several examples.

1.3 Research objectives

As we have already stated in the motivation, we focus on SPEM transformation to the Semantic Web technical space and on concrete utilizations of such approach. Since we have evaluated that the above mentioned proposals have addresses to this area only partially, our objectives are:

Objective1: to present the more comprehensive analysis of the SPEM 2.0 Metamodel and the SPEM UML Profile in order to create a SPEM OWL-DL ontology

Objective2: to create the more accurate ontology architecture for SPEM 2.0 with respect of the SPEM metamodel 2.0

Objective3: to present and implement a utilization of created SPEM ontology for a SPEM model validation

Objective4: to present and implement a utilization of created SPEM ontology for a project plan verification

Objective5: to present and implement a utilization of created SPEM ontology for a software project enactment with a supplier

1.4 Organization of the thesis

The thesis is organized into three parts: The problem (Chapter 1), the solution (Chapters 2 through 6), and the evaluation (Chapters 6 and 7).

In *Chapter 2*, we present an approach to SPEM transformation to the Semantic Web technical space. First we introduce motivation for such transformation in more detail that is followed by scope definition of the transformation. We discuss the problems that are implied from the SPEM metamodel we have addressed. Finally we present solution, the proposed SPEM Ontology. *Chapter 3* presents an approach to SPEM model validation with ontology. Since SPEM in general provides two kinds of models, a SPEM method content and a SPEM process model, we discuss several possibilities that can provide a solution for both. Based on appropriate possibility we propose an ontology based engine for a SPEM model validation. Since we intend to reuse the proposed engine for further utilizations, we also discuss the problematic with respect to them. *Chapter 4* presents an approach to project plan verification. Seeing that we have solved the method of approach in the previous chapter, we do not discuss the problems but we propose extended engine for project plan verification in an instant. Likewise as the previous chapter, in *Chapter 5* we reuse and extend the engine for SPEM model validation in order to present an approach to software project enactment with a supplier. The chapter additionally introduces usability of the SPEM Method Plugin concept in verification process. *Chapter 6* presents implementations of proposed utilizations. Finally, *Chapter 7* presents conclusion which consists of discussion of the contributions, their comparison to the related works and finally our future work.

Part II
Solution

2 Transforming SPEM to the Semantic Web Technical Space

SPEM 2.0 is used to define software and systems development processes and their components. The scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, without adding specific features for particular development domains or disciplines (e.g., project management). SPEM is a process engineering metamodel as well as conceptual framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes (OMG, 2008). The conceptual framework is depicted in Figure 2.

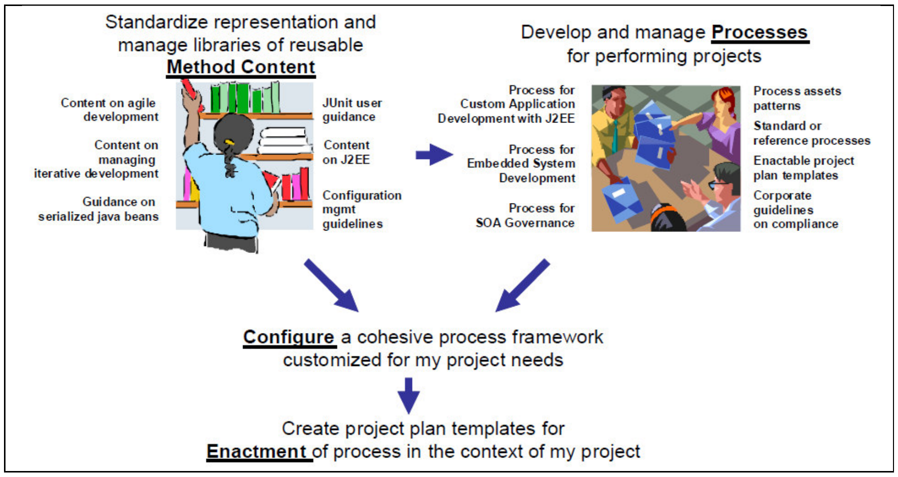

Standardize representation and manage libraries of reusable **Method Content**
Content on agile development
Content on managing iterative development
Guidance on serialized java beans
JUnit user guidance
Content on J2EE
Configuration mgmt guidelines
Develop and manage **Processes** for performing projects
Process for Custom Application Development with J2EE
Process for Embedded System Development
Process for SOA Governance
Process assets patterns
Standard or reference processes
Enactable project plan templates
Corporate guidelines on compliance
Configure a cohesive process framework customized for my project needs
Create project plan templates for **Enactment** of process in the context of my project

Figure 2. SPEM 2.0's conceptual usage framework

Since the SPEM's conceptual framework and its capabilities play the key role of the transformation, we presents them in following lists. The conceptual framework is indented for follow implementations:

- To provide a standardized representation and managed libraries of reusable method content

- To support systematic development, management, and growth of development processes
- To support deployment of just the method content and process needed by defining configurations of processes and method content:
- To support the enactment of a process for development projects

The SPEM's key capabilities are:

- Clear separation of method content definitions from the development process application of method content
- Consistent maintenance of many alternative development processes
- Many different lifecycle models
- Flexible process variability and extensibility plug-in mechanism
- Reusable process patterns of best practices for rapid process assembly
- Replaceable and reusable Process Components realizing the principles of encapsulation

2.1 Motivation

The motivation for the SPEM metamodel transformation to an OWL-DL ontology is constituted on differences between the SPEM metamodel and its possible OWL-DL representation. In the previous, we have raised the important question, which contributions can such transformation provide? To answer to this question we have to introduce the concepts *lightweight* and *heavyweight* ontology first. This distinction is a simplification of the classification based on the level of richness of their internal structure, whereby lightweight ontologies will be principally taxonomies, while heavyweight ontologies are those which model a certain knowledge "in a deeper way and provide more restrictions on domain semantics" (Goméz-Pérez, 2004). A lightweight ontology typically provides just a taxonomy of related concepts and terms, with very few cross-taxonomical links (properties), very few logical relations between the concepts, and very few axioms and constraints imposed on the concepts. Unlike formal heavyweight ontologies that are difficult to create because they include a number of properties, axioms, and constraints, lightweight ontologies are easy to create (Gašević, 2009). Since the SPEM metamodel contains semantics supplemented with natural language on the one hand and is represented with semiformal UML (Cranefield, 2001) on the other hand, it is evident that the SPEM metamodel is incapable to make statements about what can be expressed in a SPEM model with the formal system like OWL-DL ontology. Thus the SPEM metamodel is not as heavyweight as its possible SPEM OWL-DL representation which could provide expressiveness of the First Order Logic (Sowa, 2002) formal system like OWL-DL does. It is obvious that SPEM conceptual framework capabilities presented in the previous chapter can be reached rather with ontology than with a metamodel. For example, first capability is to provide a standardized representation and managed libraries of reusable method content. The purpose is to support development practitioners in setting-up a knowledge base of intellectual capital for software and systems development that would allow them to manage and deploy their content using a standardized format (OMG, 2008). We have mentioned that the semiformal SPEM metamodel is incapable to provide it with strong formal system, because it is as not heavyweight as its possible SPEM OWL-DL representation. Thus it is clear that a SPEM ontology can improve the SPEM conceptual framework, since an ontology in general clarifies the knowledge structure (Chandrasekaran, 1998), reduces conceptual and terminological ambiguity (Uschold, 1996) and allows the sharing of knowledge (Lassila, 2001). Therefore we decided for the transformation.

2.2 Subject of the transformation

The SPEM metamodel is MOF-based and reuses UML 2 Infrastructure library (OMG, 2009c). Its own extended elements are structured into seven meta-model packages. Above these packages SPEM defines three Compliance Points which are:

- SPEM Complete
- SPEM Process with Behavior and Content
- SPEM Method Content.

The scope of our solution is covered with Compliance Point "SPEM Complete". The reason of this compliance point is because we need to use all SPEM elements, where the most important are defined within the Method Content package, the Process with Method package and the Method Plugin package.

The Method Content metamodel package provides the concepts for SPEM users and organizations to build up a development knowledge base that is independent of any specific processes and development projects. The Method Content elements are the core elements of every method such as Roles, Tasks, and Work Product Definitions. Figure 3 depicts a simple method content which defines method of Requirements Specification.

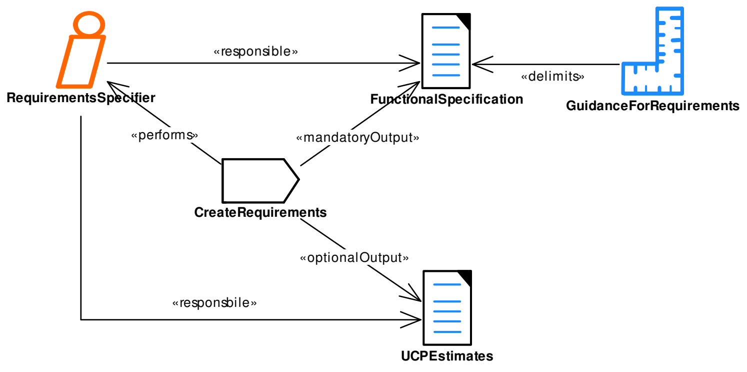


Figure 3. Example of Requirements Specification Method Content

The second necessary metamodel package that we need is the **Process with Method** metamodel package. Process with Methods defines the structured work definitions that need to be performed to develop a system, e.g., by performing a project that follows the process. Such structured work definitions delineate the work to be performed along a timeline or lifecycle and organize it in so-called breakdown structures. Figure 4 depicts a simple Requirement Process that consists of the Activity “Requirements Specification” and Activity “Prioritize Requirements”.

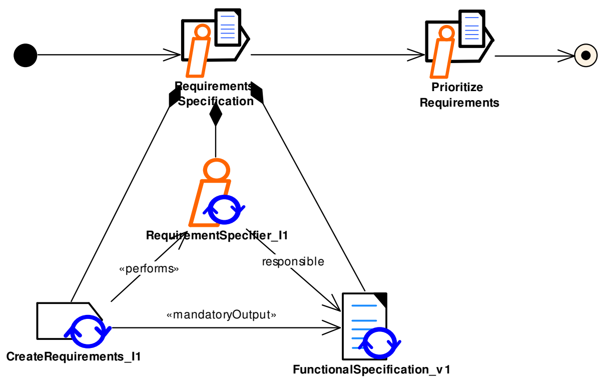


Figure 4. Example of Requirements Process

The Activity “Requirements Specification” depicted in Figure 4 does not consist of the Method Content “Requirements Specification” depicted in Figure 3, even it is logical. As we have already mentioned, one of the SPEM’s key capability is the clear separation of method content definitions from the development process. Therefore the Activity “Requirements Specification” instead of Method Content elements consists of Method Content Use elements. These elements are abstract generalization for special Breakdown Elements that reference Method Content elements. The Method Content Use elements are defined in the Process with Method metamodel package. The reason of such separation is the fact that a Method Content can be used with different Processes, such as iterative, agile etc. To be more illustrative, Figure 5 depicts an example of such Method Content separation from its Method Content Use with the relation Content Trace. As it can be seen, the figure depicts the Method Content Use “Requirements Specification” twice. First time is in an iteration 1 where just the Work Product Definition “Functional Specification” is output, and the second time in an iteration 2, where the additional output Work Product “UCP Estimated” is defined.

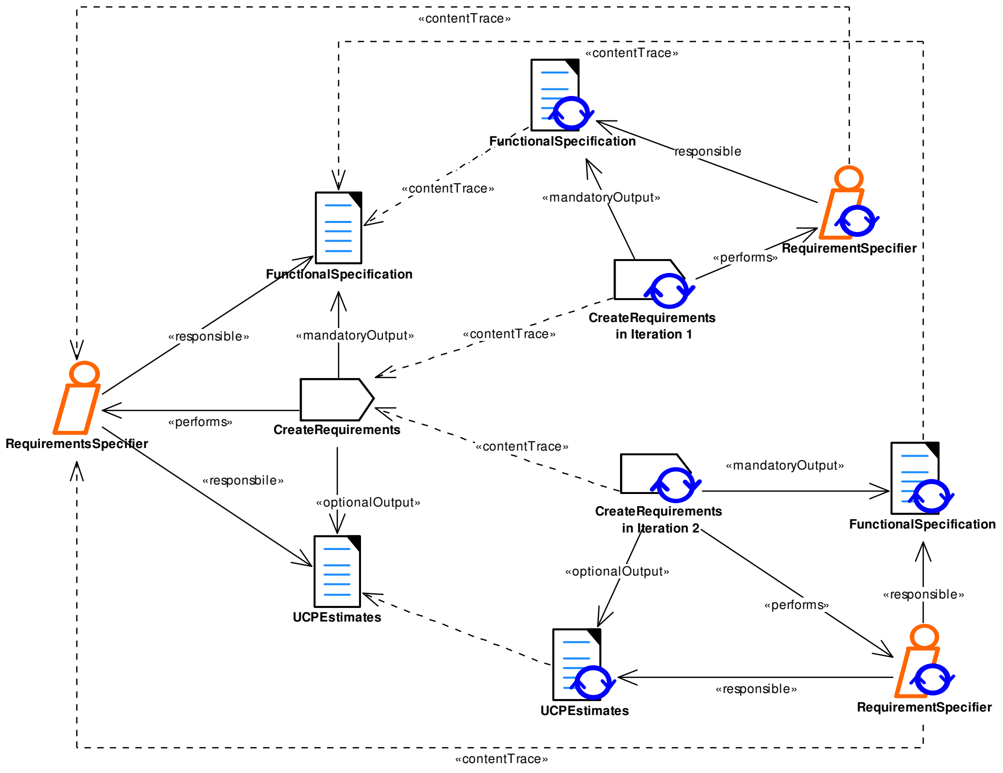


Figure 5. Example of Method Content Uses referencing Method Content

Finally, the third necessary package is the **Method Plugin** metamodel package. The Method Plugin allows extensibility and variability mechanisms for Method Content and Process specification. It provides more flexibility in defining different variants of method content and processes by allowing content and process fragments to be plugged-in on demand, thus creating tailored or specialized content only when it is required and which can be maintained as separate units worked on by distributed teams. Figure 6 depicts an example of a Method Plugin with its relationships to a Method Content and Process (method content use elements).

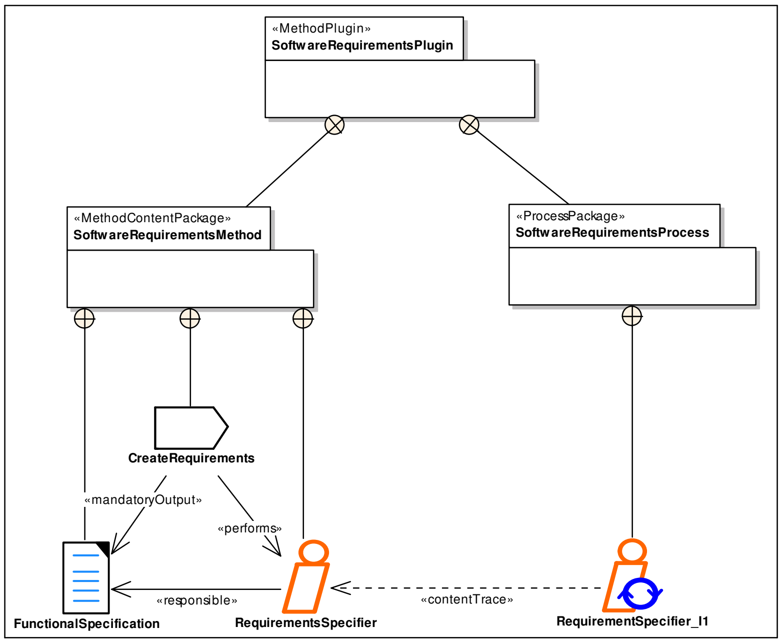


Figure 6. Example of Method Plugin

2.3 The transformation problem

Since we want to work with SPEM as with an ontology, we need to transform SPEM from the MDA Technical Space to the Semantic Web technical space to an ontology. We can use several approaches to do so (Bézivin, 2006). Certainly we preferred fully automatic approaches, but at last we concluded that this method was unrealizable. The first such approach is a XSL transformation from the SPEM metamodel serialized in XMI (OMG, 2007) directly to a SPEM ontology. Table 1 shows the mapping from the elements of the SPEM XMI serialization to the elements of the target SPEM ontology.

Table 1. Mapping between SPEM XMI and SPEM Ontology

SPEM element	OWL element
cmof:Class	owl:Class
cmof:Enumeration	owl:Class
cmof:Property	owl:objectProperty
A superclass B	B owl:subClassOf A

But even we had generated SPEM ontology classes with their taxonomy taken from the SPEM metamodel, the fact is that this approach was not sufficient, because the generated ontology does not contain all needed information. The biggest problem was to identify relationships used for a SPEM model specification, hence for object properties necessary for a SPEM ontology either. To be more concrete, Figure 7 shows an excerpt of the SPEM metamodel that defines several Method Content elements and their relationships.

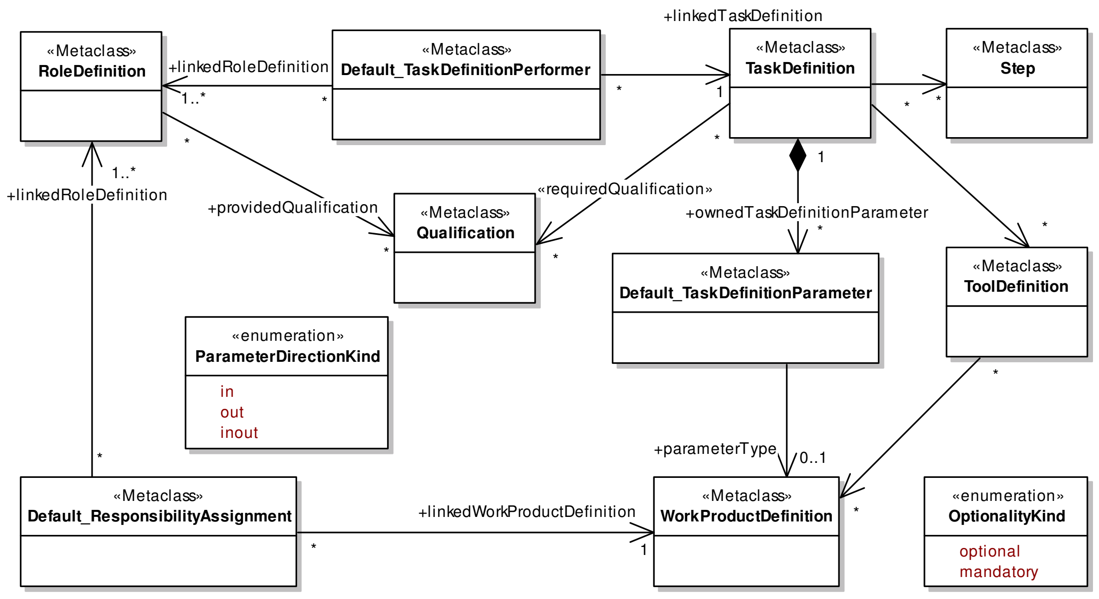

«Metaclass» RoleDefinition
+linkedRoleDefinition
1..*
«Metaclass» Default_TaskDefinitionPerformer
+linkedTaskDefinition
«Metaclass» TaskDefinition
«Metaclass» Step
+providedQualification
«Metaclass» Qualification
«requiredQualification»
+ownedTaskDefinitionParameter
«Metaclass» Default_TaskDefinitionParameter
«Metaclass» ToolDefinition
«enumeration» ParameterDirectionKind
in
out
inout
+parameterType
0..1
«Metaclass» Default_ResponsibilityAssignment
+linkedWorkProductDefinition
«Metaclass» WorkProductDefinition
«enumeration» OptionalityKind
optional
mandatory

Figure 7. Key Method Content Elements and their relationships

The figure depicts that the Default Responsibility Assignment element is a metaclass associated with the Work Product Definition metaclass and the Role Definition metaclass. But from this metamodel it is impossible to find out, that the Default Responsibility Assignment is an association between the Role Definition and the Work Product Definition in a SPEM model, as it is shown in Figure 3 for example. Thus it was impossible to transform this element to an object property in the transformed SPEM ontology. Moreover, it was also impossible to find out, what element should be the domain and the range of this relation. As it is shown in Figure 3 Requirements Specification activity contains two Default Responsibility

Assignments. First is between the Role Definition “Requirements Specifier” and the Work Product Definition “Functional Specification” and the second between the Role Definition “Requirements Specifier” and the Work Product Definition “UCP Estimates”. Both are associations, but this information was not obtainable from the SPEM metamodel.

What we need is to have SPEM in such form that defines its syntax either. To do so, we can use the XMI form of the SPEM UML Profile that fulfills exactly this requirement. This time, it was possible to identify a relationship, since such element extends the Association or Dependency metaclasses. But again, it was impossible to identify the domain and the range of a relation. Moreover, even the stereotypes are structured in the taxonomical hierarchy likewise in the SPEM metamodel, some of them are bypassed. For example, the Method Content Use stereotype is omitted, thus the Task Use stereotype is generalized from the Breakdown stereotype directly. The reason is because the SPEM UML Profile does not need the full taxonomy, because in general, the purpose of a UML profile is to provide concrete language elements to create a model rather than represent semantics (Atkinson, 2002).

However, it was finally certain, that the full automatic approach of the SPEM ontology was not possible. As we have already mentioned, some elements are missing in the SPEM UML Profile. Nevertheless, they are essential from the ontology point of view. For example, the reason can be to assert a common axiom to a parent element only once, rather than multiple times to its all child elements. The motive is to have as simplest definition of the ontology as possible, because the more simple ontology the better is its maintenance. Since the missing Method Content Use stereotype is the key concept for realizing the separation of processes from method content, it is mentioned in the SPEM specifications and in our work very often. Therefore it is very essential to have it in a SPEM ontology also, because one of the main ontology purposes is to define all essential concepts from a domain that an ontology describes.

Other problem we have addressed in the SPEM UML Profile is that its XMI serialization does not included the keywords of the stereotypes that are used for a stereotype presentation. For example, the stereotype „performs” depicted in Figure 3 on the relationships between the Task Definition “Create Requirements” and the Role Definition “Requirements Specifier” is defined as a stereotype named “Performer” in XMI. Since our objective is to use the commonly accepted concrete notation for SPEM elements, we found additional reason to abandon just automatic generation of a SPEM ontology.

2.4 Solution

Based on previous facts we have decided that our approach of a SPEM ontology creation will be semiautomatic. First, we combine previous two methods to transform the SPEM elements with their taxonomy to a SPEM ontology. We indentify a type for each element of the SPEM metamodel with the SPEM UML Profile. If the metamodel element in not a relation kind or a package kind we transform it to the ontology to an ontology class. The automatic part of our approach is depicted in Figure 8.

```
For i=1 to Count(SPEM.Metamodel.Elements) do{
    If SPEM.UMLProfile.Element.typeOf(SPEM.Metamodel.Element(i)!="association" and
                                      SPEM.Metamodel.Element(i)!="dependency"
                                      SPEM.Metamodel.Element(i)!="package")
    SPEM.Ontology.CreateOwlClass(SPEM.Metamodel.Element(i))}
```

Figure 8. Algorithm of SPEM OWL Classes generation

Note that a package metaclass element is transformed to an ontology since the purpose of both concepts is to group elements into logical units (Djurić, 2005). So, we have generated full taxonomy of the SPEM metamodel classes that constitute the SPEM ontology. Many of the OWL classes were initially abstract metaclasses, thus not aimed for a SPEM model specification, but they are essential for purposes of ontology, as we have already mentioned. The key SPEM ontology classes are depicted in Figure 9.

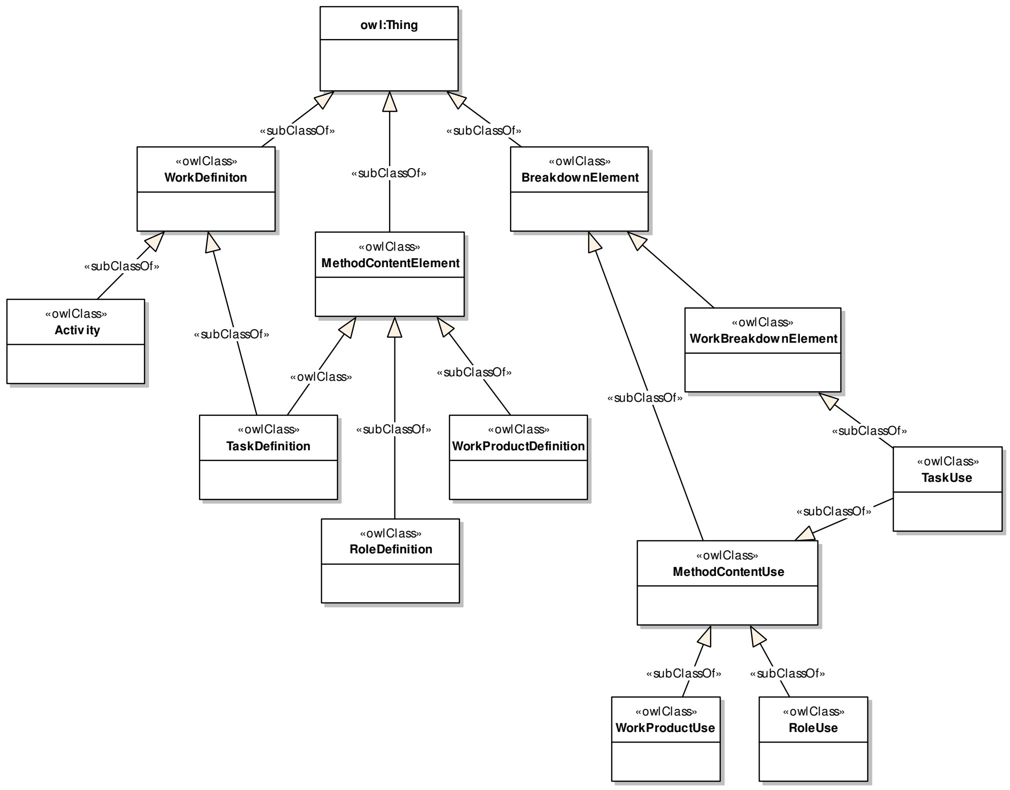

owl:Thing
«subClassOf»
«owlClass» WorkDefiniton
«owlClass» MethodContentElement
«owlClass» BreakdownElement
«owlClass» Activity
«owlClass» TaskDefinition
«owlClass» WorkProductDefinition
«owlClass» RoleDefinition
«owlClass» WorkBreakdownElement
«owlClass» TaskUse
«owlClass» MethodContentUse
«owlClass» WorkProductUse
«owlClass» RoleUse

Figure 9. The key SPEM elements represented with Ontology UML Profile

The second step that had to be done for SPEM Ontology development was to create the relationships between the ontology classes. As we have already mentioned it was not possible to do this with automatic approach, thus we have to create them manually. Table 2 depicts the mapping between the SPEM UML Profile stereotypes that are relation kind to the object properties in the ontology.

Table 2. The mapping between the SPEM relations and the object properties

SPEM Stereotype	Object Property	Domain / Range
Performer	performs	Task Definition / Role Definition
Responsibility Assignment	responsible	Role Definition / Work Product Definition
Output	output	Task Definition / Work Product Definition
Input	input	Task Definition / Work Product Definition
Content Trace	contentTrace	Method Content / Method Content Use
NestedBreakDownElement	nestedBreakDownOf	Method Content Use / Method Content Use

The table depicts also the domain and range ontology class for each object property. We have to set them manually. These key object properties with appropriate domain and range are depicted in Figure 10.

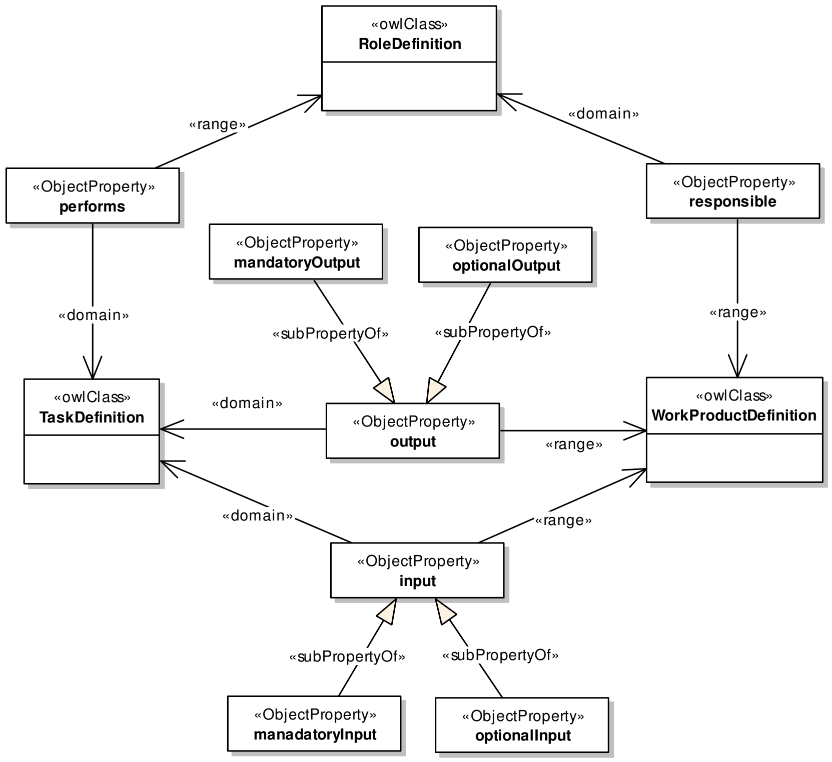


Figure 10. The key object properties of the SPEM Ontology

So, in the previous, we have presented the transformation of the SPEM metaclasses and their relationships to the SPEM Ontology. The created ontology provides the concepts for a SPEM method ontology and also for a SPEM process ontology. Thus we have transformed the Method Content metamodel package together with Process with Method metamodel package to the Semantic Web technical space. Last metamodel package that had to be transformed was the Method Plugin metamodel package. A Method Plugin is a Package that represents a physical container for Content and Process Packages. It defines a granularity level for the modularization and organization of method content and processes. A Method Plugin can extend many other Method Plugins and it can be extended by many Method Plugins. To transform it, we discuss its semantics first. Figure 11 depicts its key metaclasses in more detail.

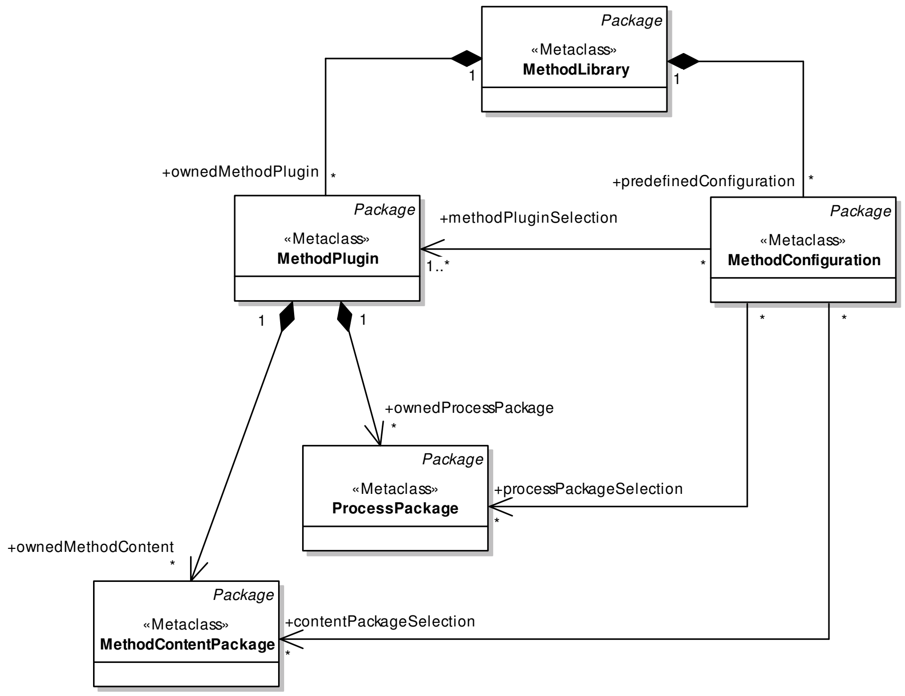


Figure 11. The key elements of the Method Plugin metamodel package

As it can been seen in Figure 11, the Method Plugin, Method Configuration, Method Library, Process Package and Method Content Package has a metaclass Package as their superclass, therefore even they are Packages. Thus the Method Content Package "Software Requirements Method" depicted in Figure 6 will be used as the Software Requirements Method Ontology, whereas the Process Package "Software Requirements Process" as the Software Requirement Process Ontology. Likewise as a package from MDA is transformed to an ontology, the Import relation in the MDA technical space is transformed to the Import relation in the Semantic Web technical space. Since even the Containment relation between packages can be represented with the Import relation between ontologies, we use this transformation either. Therefore the Method Plugin Package "Software Requirements Plugin" depicted in Figure 6 is transformed to the Software Requirements Plugin Ontology. Consequently, the ontology imports the Software Requirements Ontology and the Software Requirements Process Ontology.

Finally, we can evaluate our approach in short. Since we have merged the SPEM metamodel to the SPEM UML Profile, we have created an extended SPEM UML Profile in matter of fact.

Hence the mapping between the created Extended UML Profile and UML Ontology Profile was established which is presented in Table 3.

Table 3. Excerpt of the mapping between the SPEM Extended UML Profile and the Ontology UML Profile

SPEM UML Profile Element	UML OWL Profile
Role Definition	owlClass
Task Definition	owlClass
Role Definition	owlClass
Method Content	owlClass
Method Content Use	owlClass
performs	objectProperty
responsible	objectProperty
contentTrace	objectProperty

Since we were able to create such mapping, it was possible to represent the SPEM Ontology with the UML Ontology Profile as it is depicted in Figures 9 and 10. Since the hallmark work (Gašević, 2009) proposes the transformation of a MDA standard to the Semantic Web technical space with a mapping between UML Ontology Profile and an arbitrary UML Profile, we conclude that we created the solution that is conformable to this approach. Its use is depicted in Figure 12.

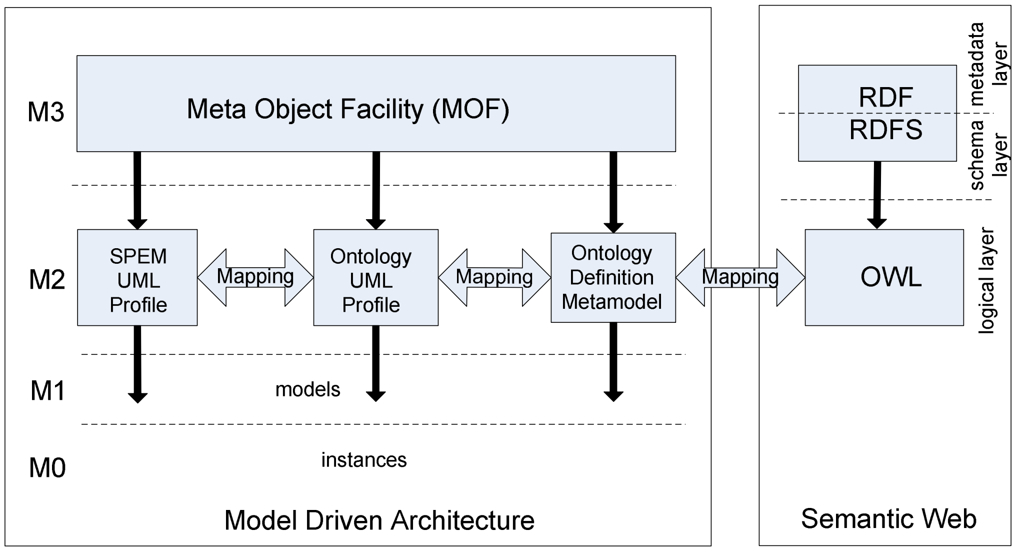


Figure 12. Mapping between SPEM and OWL

3 An Approach to Ontology Oriented SPEM Models Validation

In the previous chapter we have described the SPEM transformation from the MDA technical space to the Semantic Web technical space, where the result was the SPEM Ontology. Now, when we have the SPEM Ontology created, we are going to present its concrete first utilization. In this chapter we will use the ontology for a SPEM model validation.

Before the method presentation, it is necessary to mention the difference between the verification and the validation. Verification is an attempt to ensure that a product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities. The validation is an attempt to ensure that the right product is built, that is, the product fulfills its specific intended purpose (IEEE, 2004). In other words, we ask "Are we creating a product correctly?" if we verify a product, and we ask "Are we creating a correct product?" if we validate a product.

As the name of this chapter signs, we intended to use the SPEM ontology for a model validation. Thus a model is a product from this point of view. But how can the SPEM Ontology support a model validation? There are two main utilization scenarios, where it is essential. The first scenario is to validate a SPEM method content model and the second is to validate a SPEM process model.

Scenario 1- SPEM method model validation with ontology:
As it was already mentioned a method content model represents a model of development knowledge base that is independent of any specific processes and development projects. The example of a method content depicted in Figure 3

represents a Software Requirements Method, which defines one Role Definition “Requirements Specifier”, one Task Definition “Create Requirements” and two Work Product Definitions “Functional Specification” and “UCP Estimates”. What the model does not define is how this method will be used in the process, whether it will be iterative, agile etc. Hence what a method content model validation can be good for? This scenario is essential to ensure, that a method content is defined with the proper SPEM semantics. For example, whether a Task Definition has the proper domain of the “performs” relation that should be a Role Definition elements.

Scenario 2- SPEM process model validation with ontology:
The Method Content Use elements are the key concept for realizing the separation of processes from method content, thus a process model validation can be used to ensure, whether a process conforms to a method content definition it traces. For example Figure 5 depicts one possible method content use of the Requirement Software Method, which is depicted in Figure 3. Since this method content states that the Work Product Definition ”Functional Specification” is a mandatory output from the Task Definition “Create Requirements”, it is imperative, that this work product must be the output from every such task used in a process, e.g. in every iteration. Certainly, this validation scenario can be used likewise the first one, hence for the validation, whether a process conforms to the SPEM semantics. For example, whether a Method Content Use element references one Method Content element.

3.1 Problem

In order to enable a SPEM method content model and a SPEM process model validation, we decided to create reusable engine that will be the base engine for every our further SPEM utilizations. Since the key models of the SPEM are the method content model and the process model, the engine should work with these models inevitably. We have already presented the SPEM Ontology creation in Chapter 2, thus only a transformation of a SPEM method content model to a SPEM ontology and a SPEM process model to a SPEM process ontology has to be created. Consequently we have to present an OWL DL based engine that can reason with all these three ontologies. To accomplish this task, we have a several possibilities which we discuss in following text.

The first possibility is to use a SPEM method ontology and a SPEM process ontology as the instances of the SPEM Ontology. For example the Role Definition “Requirements Specifier” and also its concrete method content use “Requirements Specifier Use” can be the individuals of the ontology class Role Definition and Role Use from the SPEM Ontology. To be more precise we give an excerpt of such option in Figure 13.

```
SPEMOntology = { MethodContentElement, MethodContentUse, RoleDefinition,
MethodContentUse ⊑ 1 contentTrace.MethodContent, …}
SPEMMethodOntology1 = {RoleDefinition (“RequirementsSpecifier”)…}
SPEMProcessOntology1 = {RoleDefinitionUse (“RequirementsSpecifierUse”),
contentTrace (“RequirementsSpecifier”, “RequirementsSpecifierUse”) ...}
```

Figure 13. First possibility of the relationship between SPEM ontologies

The second possibility is to state a SPEM method ontology as the specialization of the SPEM Ontology and a SPEM process ontology as the instance of a SPEM method ontology. Likewise as a method content model is separated from its use in the process, the model level is separated from its instance. Thus a software process is the instance of a software method. Note that the Content Trace object property is not necessary, because is replaced by the Instantiation relation. Figure 14 depicts it formally.

```
SPEM= { RoleDefinition…}
SPEMMethodContent1 = { RequirementsSpecifier ⊑ RoleDefinition, …}
SPEMProcess1 = { RequirementsSpecifier (“RequirementsSpecifierUse”) …}
```

Figure 14. Second possibility of the relationship between SPEM ontologies

The third possibility depicted in Figure 15 states a SPEM method content as the instance of the SPEM Ontology, and a SPEM process ontology as the instance of a SPEM method ontology.

```
SPEMOntology = { RoleDefinition…}
SPEMMethodOntology1 = { RoleDefinition (“RequirementsSpecifier”), …}
SPEMProcessOntology1 = { RequirementsSpecifier („RequirementsSpecifierUse”) …}
```

Figure 15. Third possibility of the relationship between SPEM ontologies

However, all of these three possibilities are technically correct, but all of them are unsuitable for purposes of this work. The first problem is that we want to use OWL DL reasoning, because this dialect of OWL retains computational completeness, i.e. all conclusions are guaranteed to be computable and decidability, i.e. all computations will finish in finite time (McGuiness, 2004). Therefore the third possibility that we have mentioned must be excluded. The reason is that OWL DL does not accept that an individual can be also a class, like the Requirements Specifier is.

But why even the two others proposed possibilities are unsuitable? The reason is because both of them do not count with real individuals of a real development project. It is evident that a requirement specifier is a real person, “Samuel Fox” for example, than the Role Use “Requirements Specifier Use”. By other words, an additional instance level for real individuals is needed, but it will cause the unconformity with the OWL DL reasoning again. To avoid this problem, we can state that a SPEM process model represents the instance level, rather than class level. Formal representation of such solution is depicted in Figure 16.

SPEM={RoleDefinition…}
SPEMMethodOntology1 = { RequirementsSpecifier $\sqsubseteq$ RoleDefinition}
SPEMProcessOntoogy1 = { RequirementsSpecifier (“RoleUse”),
RequirementsSpecifier (“SamuelFox”), sameIndividualAs (“SamuelFox”, “RoleUse”) ...}

Figure 16. Fourth possibility of the relationship between SPEM ontologies

But even this solution was insufficient. The first problem with is that a SPEM process model is not intended to represent concrete individuals, thus the change of the SPEM metamodel should be made to redefine the SPEM process semantics in order to use this solution. Moreover in a real project, concrete resources (e.g. “Samuel Fox”) are assigned much latter than an appropriate SPEM process is selected. Hence even this solution is unusable.

3.2 Solution

Based on previous facts we have realized we need a solution that addresses to all mentioned problems. Solution must to enable to validate the SPEM Ontology, a

SPEM method ontology and a SPEM process ontology. It also must be extendable with the individuals from real projects (e.g. obtainable from a project plan) and finally, the solution must be OWL DL conformable.

An approach that fulfills to all these requirements is that a SPEM method ontology will be the specialization of the SPEM Ontology and a SPEM process ontology will be the specialization of the SPEM Ontology either. To be more demonstrative Figure 17 depicts an excerpt this approach.

```
SPEM={ MethodContentElement, MethodContentUse, RoleDefinition, RoleUse,
MethodContentUse ⊑ 1 contentTrace.MethodContent, …}
SPEMMethodContent1 = { RequirementsSpecifier ⊑ RoleDefinition … }
SPEMProcess1 = { RequirementsSpecifierUse ⊑ RoleUse,
contentTrace(RequirementsSpecifierUse, RequirementsSpecifier }
```

Figure 17. The final solution of the relationship between SPEM ontologies

Figure 18 depicts the presented solution from the MDA and the Semantic Web point of view. It shows the mapping between a SPEM metamodel element and a SPEM ontology class, between a SPEM method content model element and a SPEM method ontology class and finally between a SPEM method content use model element and a SPEM method content use ontology class.

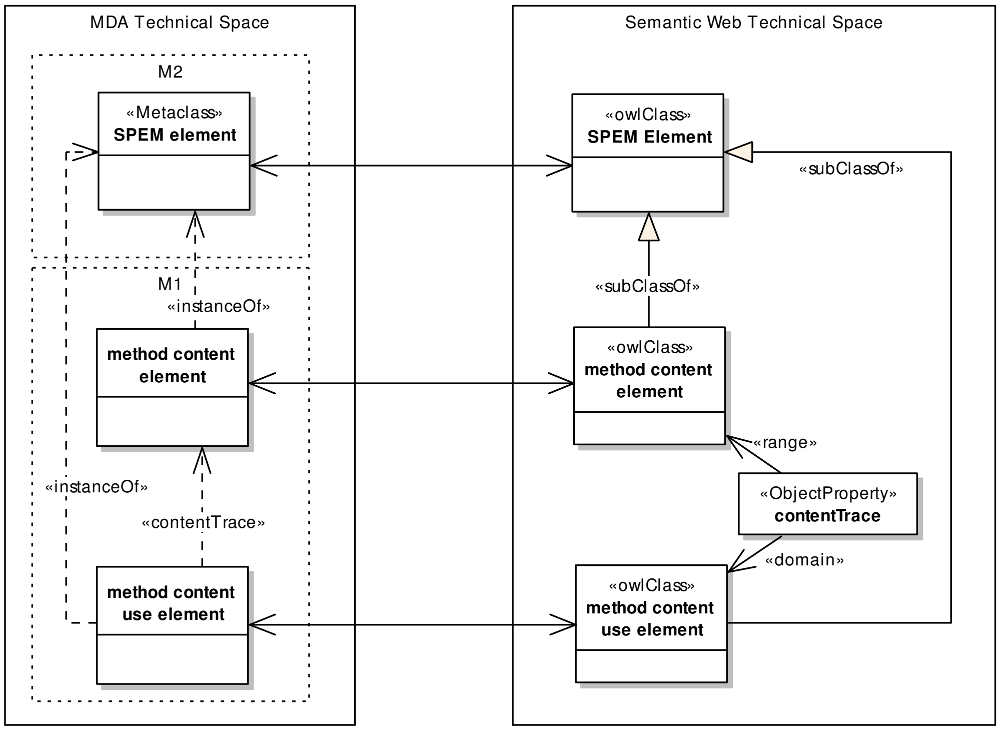


Figure 18. Mapping between elements of SPEM in MDA and SPEM in the Semantic Web in the context of the Approach to SPEM Models Validation with Ontology.

Since we have resolved the problems that we have stated, we can continue to define the engine for a SPEM model validation. As we have already mentioned the engine should be usable for two scenarios, a SPEM method content model validation and a SPEM process model validation. To do so, the mandatory condition is to transform both models from the MDA technical space to the Semantic Web technical space. This requirement is not hard to accomplish because an UML model transformation even extended with arbitrary UML Profile can be transformed to a XMI serialization (Kleppe, 2003). All we need is to create a transformation from a serialized SPEM method content model and from serialized SPEM process model to the SPEM method ontology and SPEM process ontology. Due to these requirements, we have created XSL transformations *SPEMMethodContent2OWL* and *SPEMProcess2OWL*. The former transforms a method content model serialized in XMI to a SPEM method ontology, and the latter transforms a SPEM process model to a SPEM process ontology. Since we have created the SPEM Ontology already, the OWL DL reasoning can be executed with an OWL DL reasoner. If the reasoning process contains only the SPEM Ontology and a SPEM method ontology, then the first SPEM model validation scenario is accomplished, or if a SPEM process ontology is additionally included, then the second SPEM model validation scenario can be executed. If the reasoner finds the inconsistency, its source should be removed. The engine for a SPEM model validation is depicted in Figure 19.

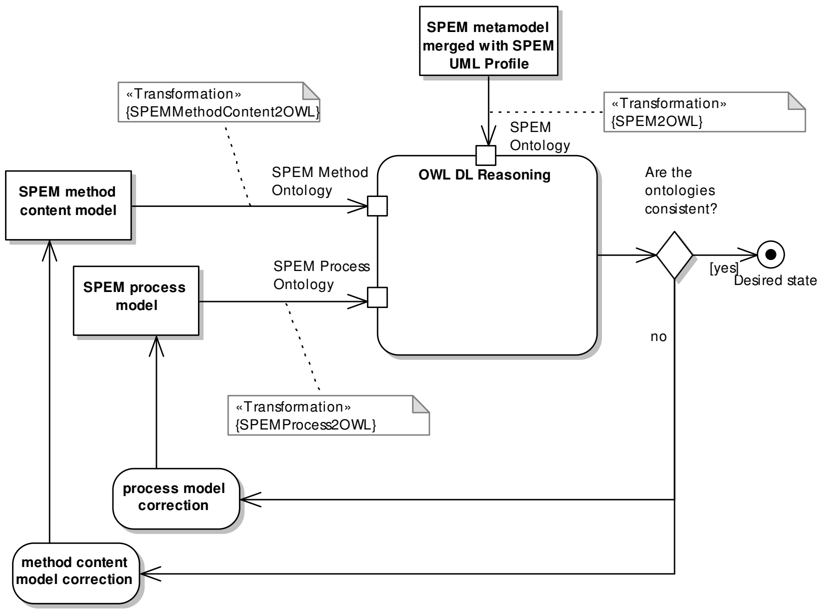

SPEM metamodel merged with SPEM UML Profile
«Transformation» {SPEMMethodContent2OWL}
«Transformation» {SPEM2OWL}
SPEM Ontology
SPEM Method Ontology
OWL DL Reasoning
Are the ontologies consistent?
SPEM method content model
SPEM Process Ontology
SPEM process model
[yes]
Desired state
no
«Transformation» {SPEMProcess2OWL}
process model correction
method content model correction

Figure 19. The Approach to Ontology Oriented SPEM Models Validation

To be more precise, we give formally defined conditions that cover both validation scenarios. Formula 1 address to the first validation scenario, which is the a SPEM method model validation with ontology and Formula 2 address the second, which is a SPEM process model validation with ontology. We say, that a SPEM method model is consistent with the SPEM ontology if it is true that

$$\text{SPEM Ontology} \vdash \text{SPEM method ontology} \tag{1}$$

Likewise we say that a SPEM process model is consistent with a SPEM method ontology and the SPEM Ontology if it true that

$$\text{SPEM Ontology} \vdash \text{SPEM method ontology} \vdash \text{SPEM process ontology} \tag{2}$$

4 An Approach to Ontology Oriented Project Planning

Software project management is the art of balancing competing objectives, managing risk, and overcoming constraints to deliver a product that meets the needs of the customers and the end users (Kruchten, 2003). Project management is accomplished through the use of processes such as: initiating, planning, executing, controlling and closing [2]. How the project will be managed and how the plan will be managed must also be planned. Reporting, monitoring, and control of the project must fit the selected software engineering process and the realities of the project, and must be reflected in the various artifacts that will be used for managing it. But, in an environment where change is an expectation rather than a shock, it is vital that plans are themselves managed. This requires that adherence to plans be systematically directed, monitored, reviewed, reported, and, where appropriate, revised (IEEE, 2004). To support these general objectives, we present an ontology based approach to project planning. We discuss two scenarios that could support a project planning. So, the third scenario presented in this thesis is a project plan creation with ontology scenario and the fourth is the scenario of a project plan verification with ontology.

Scenario 3 - Project plan creation with ontology:
Since the SPEM allows enacting a project planning system with the Method Content Use elements, the XSL transformation from a SPEM process model to a project plan can be executed; therefore it is no need to use OWL DL reasoning for a project plan creation. On the other hand, before such transformation a project manager can use ontology based reasoning with the source of the transformation, which is a SPEM process model. But this verification is covered with already presented scenario 2 in this thesis. However, a knowledge engineer can use OWL DL reasoning for generated project plan from a SPEM process model, but this scenario is rather appropriate to debug XSL transformation than a project plan. But there is a situation when the ontology based process plan creation is essential. When a project manager

wants to use more than one method content or process as a source, then the OWL DL reasoning is essential. Since this is the subject of our third utilization of this thesis, we discuss this scenario in more detail in Chapter 5 that presents the approach to project enactment with a supplier.

Scenario 4 - Project plan verification with ontology:
The main utilization scenario that is presented is the ontology based project plan verification. This scenario is essential when a project manager wants to ensure that his project plan is according to selected method content and process. It usability is essential after any change in a project plan is made.

4.1 Solution

In the previous chapter we have presented the engine for a SPEM model validation. We have discussed several approaches to do so and chosen the best that fit to our needs. The problem was to create such solution that will comply with the SPEM architecture, such as separation of a method content from its process specification and moreover to use OWL DL reasoning. We have presented the mappings between the SPEM elements from MDA technical space to SPEM elements in the Semantic Web technical space in Figure 18, together with the architecture of the engine in Figure 19.

Our approach of the ontology based project plan verification fully reuses the previously presented ontology based SPEM model validation. The main extension is that the individuals from a project plan are also included for the reasoning process. A project planning system usually supports creation of a project plan using the breakdown elements, such as phases, milestones, tasks, roles etc. The purpose is to decompose them to the atomic elements that can be executed by or assigned to a project member. This is the reason, why the SPEM Method Content Use metaclasses are specialized from the Breakdown metaclass. The child classes of the Method Content Use metaclass are the Work Product Use, the Role Use and the Task Use. Therefore the enactment of a project plan with a SPEM process definition has to create the instantiation relation between method content uses elements included in a project plan and method content uses elements that constitute a SPEM process ontology. Thus, our approach is fully conformable with the SPEM architecture that explicitly states a process enactment with the project planning systems through the instantiation relation. From the ontology point of view the elements obtained from a project plan are the individuals of a SPEM process ontology. The Figure 20 depicts an excerpt demonstrates the main principle of the ontology based project plan verification.

```
SPEM = { TaskDefinition, TaskUse…}
SPEMMethodOntology1 = { CreateRequirements ≡ TaskDefinition… }
SPEMProcessOntology1 = { CreateRequirements_Iteration1 ≡ RoleUse,
CreateRequirements_Iteration1 ≡ ∀performs.RequirementsSpecifier_Iteration1…}

ProjectPlan1 =
{ RequirementsSpecifier_Iteration1 ("Samuel Fox"),
CreateRequirements_Iteration1 ("Create requirements in iteration 1"),
performs("Create requirements in iteration 1"," Samuel Fox") …}
```

Figure 20. Excerpt of relationship between SPEM ontologies and a project plan

Since the Role Use is the subclass of the Method Content Use, and the Method Content Use is the subclass of the Breakdown element, it can be computed that individual "Samuel Fox" is also individual of the Method Content Use and Breakdown element either. Based on previous facts it is possible to create the engine for the ontology based project verification. Since the scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, the SPEM metamodel does not include elements such as Iteration, Phase etc. The reason is because not every software development process needs to have iterations for example. Therefore we had to extend the engine for a project plan validation with the SPEM Base Plugin. This plugin is included in the SPEM specification. It provides commonly used concepts for the domain of software engineering such as Phase, Iteration, Checklist etc. We need to use these concepts to define iterative kind of a process for a method content. Note, that the SPEM Base Plugin ontology is also essential for Scenario 2 mentioned in this thesis.

So, the previous engine for a SPEM model validation is reused and simply extended with individuals, which can be obtained from a project plan. The engine that supports the ontology based project generation and verification is depicted in Figure 21.

The figure includes several additions than the engine for a SPEM model validation. First addition is a XSL transformation MPP2OWL that generates SPEM process instances that constitute project plan ontology from a XML format of a project plan. Second addition is the XSL transformation SPEMProcess2OWL that transforms XML format of a project plan to a

SPEM process model. Note that this transformation could be bidirectional, thus it is possible to generate project plan from a SPEM process model also. The last supplement is additional ontology SPEM Base Plugin, which provides concepts such as Iteration, Phase as we have already discussed.

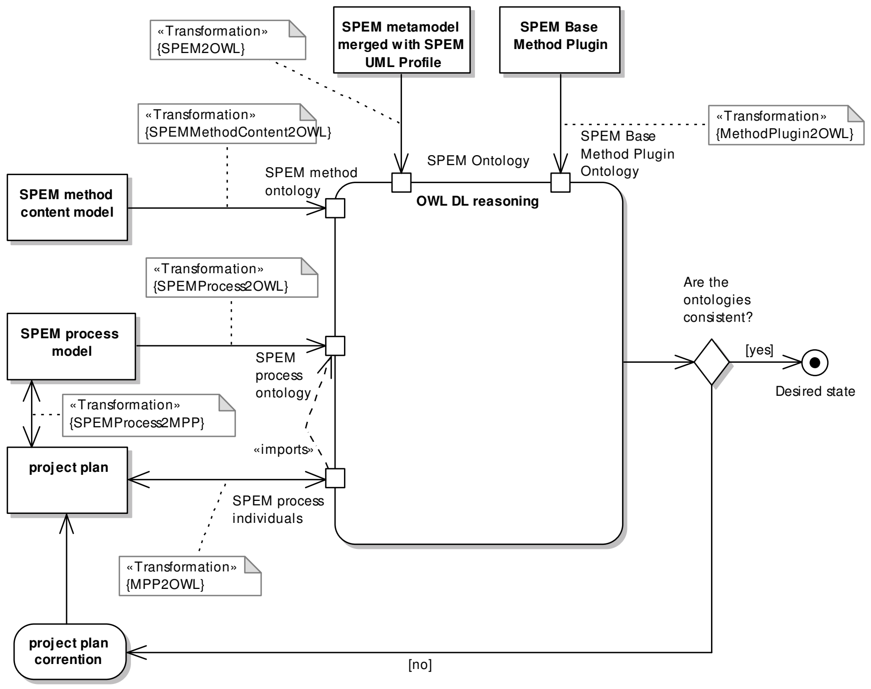

«Transformation» {SPEM2OWL}
SPEM metamodel merged with SPEM UML Profile
SPEM Base Method Plugin
«Transformation» {SPEMMethodContent2OWL}
«Transformation» {MethodPlugin2OWL}
SPEM Ontology
SPEM Base Method Plugin Ontology
SPEM method ontology
SPEM method content model
OWL DL reasoning
«Transformation» {SPEMProcess2OWL}
Are the ontologies consistent?
SPEM process model
SPEM process ontology
[yes]
Desired state
«Transformation» {SPEMProcess2MPP}
«imports»
project plan
SPEM process individuals
«Transformation» {MPP2OWL}
project plan corrention
[no]

Figure 21. The engine for ontology based project planning

To be as most demonstrative as possible we give the mapping between elements from the MDA technical space to the elements in the Semantic Web technical space in Figure 22. In comparing with Figure 18, the additional elements are the individuals of a project plan.

Note, that this approach has additionally confirmed the purpose of SPEM transformation to the Semantic Web technical space. As we have mentioned, a metamodel prescribes reality, hence a reality is constructed according to a model, whereas an ontology is descriptive, thus a reality is not constructed from it. By other words, in a descriptive model truth lies in reality, whereas in a prescriptive model, truth lies in the model itself (Favre, 2005). Even a SPEM Method content or process is rather prescriptive that, their instances, i.e. individuals of a SPEM process are rather descriptive on the contrary. Even we prescribe a software process; it is well-known fact that is it almost impossible to finish a project in time, quality and budget without any changes. The reality generates too many unexpected variables for a project, which should to adapt them rather than to adhere to original method or process.

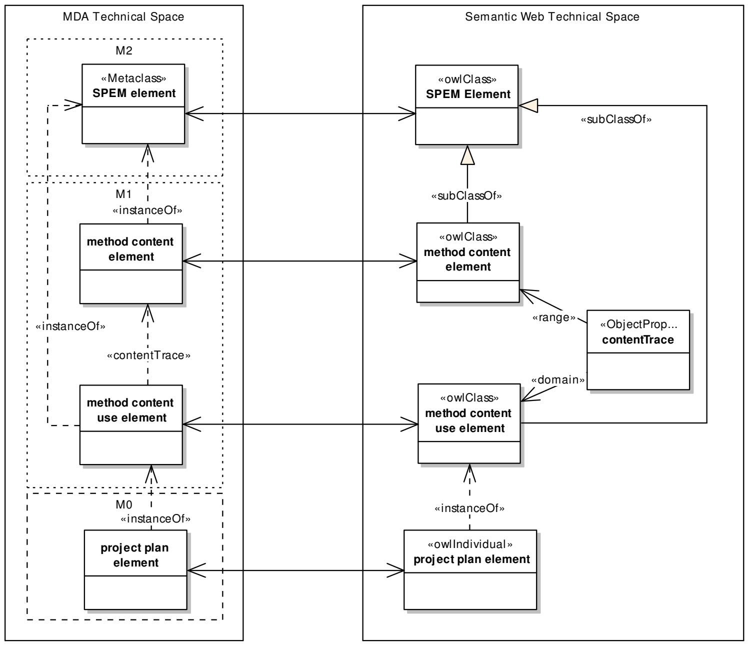


Figure 22. Mapping between elements of SPEM in MDA and SPEM in the Semantic Web in the context of the approach to ontology based project planning

However, to be more precise, we give the formally defined conditions that cover the mentioned utilization scenarios in this approach. Since we have stated that the scenario of the project plan generation will be discussed in the chapter 5, we present only formal conditions that cover the project plan verification scenario. First we define the Project Plan Knowledge as the union of the SPEM Ontology, SPEM Base Plugin Ontology, a SPEM method ontology and a SPEM process ontology, as it is shown in Formula 3.

$$\text{Project Planning Knowledge} = \text{SPEM Ontology} \sqcup \text{SPEM Base Plugin Ontology} \sqcup \text{SPEM method content ontology} \sqcup \text{SPEM process ontology} \quad (3)$$

Then we say, that the Project Planning Knowledge is satisfied in a project plan if it is true that

$$\text{Project Planning} \models \text{Software Knowledge.} \quad (4)$$

From the First Order Logic point of view, the Project Plan Knowledge is the theory and the project plan is its model. Since a theory can have a model only if a theory is consistent (Kvasnička, 2005), it is necessary, that Formula 5 is true.

$$\text{SPEM Ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{SPEM method content ontology} \vdash \text{SPEM process ontology} \quad (5)$$

5 An Approach to Software Project Enactment with a Supplier

The difficulty of software development is greatly enhanced when it is inevitable to cooperate with a supplier. The general issue is to manage a lot of differences such as different tasks, software work products, guidelines, roles etc (IEEE, 2004). The ideal state is that a company and its supplier use the same software framework and they use it equally. Otherwise risk of budget and time overrun together with quality fall is greatly increased. Unfortunately, such an ideal state cannot exist. Either companies use different software frameworks, or they use the same software framework - but necessarily differently. It is natural that companies have different knowledge obtained from their various projects, and also have different peoples with different experiences. Thus even they use the same software framework, e.g. RUP (Kruchten, 2003) the project enactment with the supplier, due to mentioned differences, is problematic.

Even if these mentioned problems seem to complex, we can address them with our previous approaches. We reuse both of them and additionally we include also the Method Plugin metamodel package concept to improve the approach's architecture. The package Method Plugin adds the capabilities of modularization and extensibility of Method Content and Processes to SPEM 2.0. In other words, it supports tailoring of Method Content and Processes without directly modifying them, but by describing changes from a separate unit called a Method Plugin.

In the follow text, we discuss a short example that helps us to explain the use of the Method Plugin. Figure 6 depicts simple Method Plugin “Software Requirements Plugin” that consists of the Method Content “Software Requirements Method” and the Process “Software Requirements Process”. Figure 5 depicts details of the Software Requirements Process that consists of Activity “Create Requirements” and “Prioritize Requirements” together with the mappings between their method contents and method content uses elements. As it can be seen, the Method Plugin “Software Requirements Plugin” is the iterative process for the requirements specification. If a project manager needs to include software requirements activities in his project, he can simply use our method plugin and then selects the elements he needs. Likewise he can use other method plugin, the Rational Unified Process Plugin for example, for other project activities such as design, testing etc. This time he avoids RUP’s requirements method content and process, because he trusts to our method and process (our method plugin) to handle requirements. So the purpose of the Method Plugin is to provide reusable method content and process for the project method and process customization. As it is shown in Figure 11, the Method Plugin metamodel element consists of the Method Content Package and Process Package. The way that Method Plugin can be configured is enabled through the Method Configuration metamodel element that selects desired Method Content Packages and desired Process Packages. Thus a Method Configuration is the feature that enables to customization of Method Plugins to create desired method content and process.

5.1 Solution

Our approach to software process enactment with a supplier is based on OWL DL verification with set of different method plugins, which represent different methods and processes of a company and its supplier. When the result of the OWL DL verification is inconsistency it means, that a project cannot be enacted with a supplier and the source of inconsistency should be removed. Therefore the necessary condition to use this method is to have company’s and supplier’s software process specified with SPEM models. Additionally, this approach can be used to other scenario than the software project enactment with a supplier that is the precise evaluation of the relationships between standard software processes such as RUP with Microsoft Solution Framework for example. Even this is out of scope of this work we wanted to point on it, since it is very interesting topic. However, the engine for a project enactment with a supplier is depicted in Figure 23.

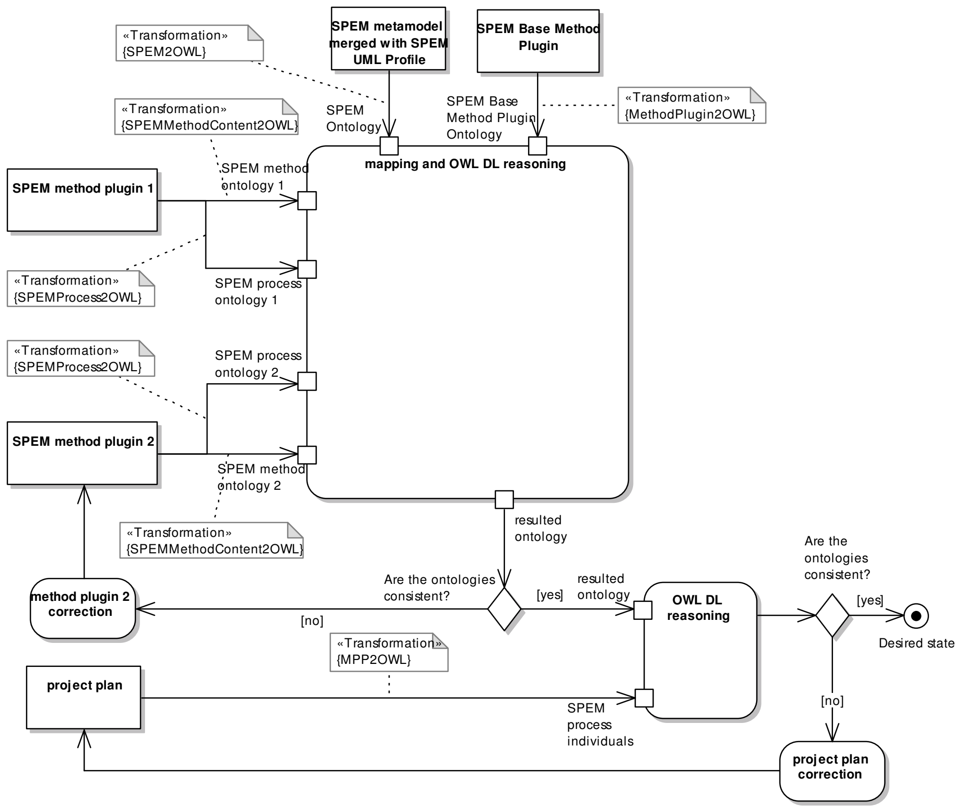

«Transformation» {SPEM2OWL}
SPEM metamodel merged with SPEM UML Profile
SPEM Base Method Plugin
«Transformation» {SPEMMethodContent2OWL}
SPEM Ontology
SPEM Base Method Plugin Ontology
«Transformation» {MethodPlugin2OWL}
mapping and OWL DL reasoning
SPEM method plugin 1
SPEM method ontology 1
«Transformation» {SPEMProcess2OWL}
SPEM process ontology 1
«Transformation» {SPEMProcess2OWL}
SPEM process ontology 2
SPEM method plugin 2
SPEM method ontology 2
«Transformation» {SPEMMethodContent2OWL}
resulted ontology
method plugin 2 correction
Are the ontologies consistent?
[no]
[yes]
resulted ontology
OWL DL reasoning
Are the ontologies consistent?
[yes]
Desired state
«Transformation» {MPP2OWL}
project plan
SPEM process individuals
[no]
project plan correction

Figure 23. The Approach to Software Project Enactment with a Supplier

The approach consists of several major steps. First it is necessary to transform the both method plugins to the ontologies. Since a Method Plugin is constituted with a Method Content and a Process we can reuse our previously defined XSL transformations *SPEMMethodContent2OWL* and *SPEMProcess2OWL* to create desired ontologies. A method plugin 1, i.e. company's method plugin is transformed to a SPEM method content ontology 1 and a SPEM process ontology 1, whereas a method plugin 2, i.e. supplier's method plugin is transformed to a SPEM method content ontology 2 and a SPEM process ontology 2. Second it is necessary to create mapping (Shvaiko, 2007) between the elements of these ontologies, because some of them are usually related. For example a Work Product Definition "Requirement Specification" from a SPEM method ontology 1 can be equivalent with a Work Product Definition "Use Case Specification" from the second method content ontology. To do this mapping the relationships such as sameClassAs, subClassOf, isEquivalentWith, subPropertyOf or samePropertyAs can be used. Third the OWL DL validation is executed to verify consistency between the both method plugins.

The following text presents several utilization scenarios of our approach, which extend the overall set of utilization scenarios mentioned in this thesis.

Scenario 5 – Verification of the set of SPEM methods with ontology:
This scenario can be used when it is necessary to verify, whether at least two different SPEM method contents are consistent. Therefore its use for the software project enactment with supplier is appropriate. Since it is necessary to manage a lot of differences such as different tasks, software work products, guidelines, roles etc., this scenario can be used to reveal and to remove such differences that are inconsistent. For example a company can state that the Task Definitions "Create Requirements" and "Create Test Cases" should not be performed by the same person, because the creation of the Test Cases can reveal hidden inconsistencies that the author of requirements does not need to be aware of. On the other hand, a supplier's method can state that the same person can perform both task definitions. Hence, this is inconsistency and it should be removed.

Scenario 6 – Verification of the set of SPEM processes with ontology:
This scenario is similar than the previous, but this time processes of a software development are the subject for the verification. It is used to verify, whether at least two different SPEM processes are consistent. For example, a company's method content requires that the Task Definition "Create Requirements" should be executes at least in two iterations to increase the quality of requirements, but supplier permits only one iteration. Again, this is inconsistency and it should be removed.

Scenario 7 – Project plan generation with the set of method plugins with ontology: This scenario can be executed when a project manager wants to create a project plan that is based at least on two method plugins. Even the ontology plays intermediate task of such scenario, its usability is crucial. First it is necessary to select the desired method contents and a processes from the set of method plugins a transform them to ontologies. Then the scenario 5 and 6 are executed to reveal inconsistencies. If the ontologies are consistent, then the XSL based transformation to XML format of the project plan can be executed. Then it sure the resulted project plan is consistent with desired method plugins.

Scenario 8 – Project plan verification the set of method plugins with ontology: This scenario can be executed when a project manager wants to verify a project plan with a set of method plugins. The scenario has the same architecture than the scenario than the project plan verification with ontology scenario. The only difference is the count of method contents and processes, which create the knowledge about project planning. When the mapping between the set of method plugins are created and their consistency is established, the project verification can be executed against these method plugins.

To be more precise, we give the formally defined conditions that cover the mentioned utilization scenarios. Since the scenario 7 consists of the scenario 5 a 6 we only present the formal specification of scenario 5, 6 and 8. Scenario 5 is covered with Formula 9, scenario 6 with Formula 10 and scenario 8 with Formula 8 and 11. First we define the two method plugins and the Project Planning Knowledge:

$$\text{SPEM method plugin 1 ontology} = \text{SPEM method ontology 1} \sqcup \text{SPEM process ontology 1} \tag{6}$$

$$\text{SPEM method plugin 2 ontology} = \text{SPEM method ontology 2} \sqcup \text{SPEM process ontology 2} \tag{7}$$

$$\text{Project Planning Knowledge} = \text{SPEM Ontology} \sqcup \text{SPEM Base Plugin Ontology} \sqcup \text{SPEM method plugin 1 ontology} \sqcup \text{SPEM method plugin 2 ontology} \tag{8}$$

Then we say the two SPEM method contents are consistent if:

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{SPEM method ontology 1} \vdash \text{SPEM method ontology 2} \tag{9}$$

and the two SPEM processes are consistent if:

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{SPEM method ontology 1} \vdash \text{SPEM method ontology 2} \vdash \text{SPEM process ontology 1} \vdash \text{SPEM process ontology 2} \tag{10}$$

The Project Planning Knowledge is satisfied in a project plan if:

$$\text{Project Plan} \models \text{Project Planning Knowledge} \tag{11}$$

Again, since a theory can have a model only if the theory is consistent, the necessary condition that enables Formula 11 to be true is that either Formula 10 must be true.

Part III
Evaluation

6 Implementation

In this chapter, we present an implementation of our previously defined approaches. Ontologies rely on well-defined and semantically powerful concepts in artificial intelligence (Návrat, 2002), such as description logics, reasoning, and rule-based systems (Viannu, 1997). Since we use OWL DL form of ontology, the implementation has goal to present the proposed utilization scenarios with a Knowledge Representation System that supports description logics. Developing a knowledge base using a description logic language means setting up a terminology (the vocabulary of the application domain) in a part of the knowledge base called the TBox, and assertions about named individuals (using the vocabulary from the TBox) in a part of the knowledge base called the ABox (Baader, 2004). The vocabulary consists of concepts and roles. Concepts denote sets of individuals. Figure 24 depicts the base architecture of a Knowledge Representation System.

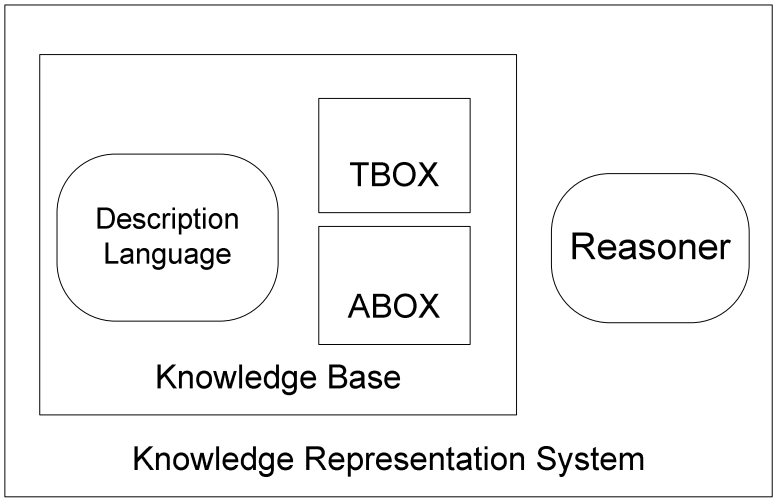


Figure 24. Knowledge Representation System

Since we want to present the implementation in the context of description logics based reasoning (Krdžavac, 2009), we either use a description logic language instead of other possible languages such as XML or functional syntax of OWL. In fact, the TBox defines (in a general way) semantic relationships between individuals introduced in the ABox and their properties. In other words, the ABox describes a specific state of affairs in the world in terms of the concepts and roles defined in the TBox (Gašević, 2009). Therefore a mapping between a Knowledge Based Representation system and the ontologies that constitute the architecture of our approach can be created. We give the mapping in Table 4.

Table 4. Mapping between components of a knowledge based representation system to our approach's ontologies

Ontology type	KBRS component
SPEM Ontology	TBox
SPEM Base Method Plugin	TBox
SPEM method content ontology	TBox
SPEM process ontology	TBox
SPEM method plugin ontology	TBox
Individual of a SPEM process ontology	ABox

As it is shown in Figure 24, a reasoner is the component responsible for the standard reasoning processes such as consistency checking, concept satisfiability, classification and realization. The consistency checking ensures that an ontology does not contain any contradictory facts. The concept satisfiability determines whether it is possible for a class to have any instances. If a class is unsatisfiable, then defining an instance of that class will cause the whole ontology to be inconsistent. The classification computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all or only the direct subclasses of a class. The realization finds the most specific classes that an individual belongs to; i.e., realization computes the direct types for each of the individuals. Realization can only be performed after classification since direct types are defined with respect to a class hierarchy (Sirin, 2007).

We have used the Protégé (Rector, 2006), a free open source ontology editor and knowledge-base framework with the Pellet (Sirin, 2007), an open source OWL DL reasoner that supports all the previously mentioned standard reasoning capabilities. These capabilities were necessary for our approaches implementation which we have created. The consistency reasoning was used to verify whether the ontologies used in our approaches are consistent or not. The concept satisfiability was helpful to ensure that created SPEM content or process model can be really used. Seeing that the unsatisfied class cannot have an individual, it means that a reality based on a SPEM model cannot exist. The classification was also very important because a ontology class must to adhere to all axioms stated with its superclasses. Finally either the realization was necessary, due to ensure that an individual adheres to all axioms stated with its all classifiers on the one hand, and that an individual has inferred just its desired classifiers on the other hand. To be more demonstrative we discuss these inference processes during concrete examples.

So, we have presented three utilizations of transformed SPEM to the Semantic Web technical space. The follow chapters describe them. Since we reuse some of the ontologies in other examples, we present them first. Note that we use the same names of ontologies depicted in follow examples as the names of the ontologies attached to this work.

spem ontology

owlClasses:
BreakDownElement, MethodContentElement, WorkDefinition, RoleDefintion, TaskDefinition, WorkProductDefinition, MethodContentUse, RoleUse, TaskUse, WorkProductUse, Activity…

objectProperties:
performs, mandatoryOutput, optionalOutput, responsible, predecessor, contentTrace, nestedBreakDownOf, delimits …

asserted axioms:
MethodContentElement $\sqcap$ MethodContentUse $\equiv \emptyset$
MethodContentUse $\sqsubseteq$ MethodContentElement $\sqcap$ $\forall$contentTrace.MethodContentElement
RoleDefintion $\sqsubseteq$ MethodContentElement $\sqcap \forall$responsible.WorkProductDefinition,
TaskDefinition $\sqsubseteq$ MethodContentElement $\sqcap \forall$mandatoryOuput.WorkProductDefinition …

Figure 25. Excerpt of the SPEM Ontology defined with a description logic

6.1 SPEM model validation with ontology

The approach to SPEM model validation with ontology is based on reasoning with the SPEM Ontology, a SPEM method ontology and a SPEM process ontology. It provides two validation scenarios. SPEM method content model validation with ontology is the first scenario and a SPEM process model validation with ontology is the second. The former is indented for situations where the Work Definitions (i.e. method contents) are the subjects for the verification and the latter for the process verification. Figure 26 depicts an excerpt of formal model of Requirements Specification Method Content depicted in Figure 3. We use the name "requirements-method ontology" to reference it.

requirements-method ontology

imports:
spem

owlClasses:
RequirementsSpecifier, CreateRequirements, FunctionalSpecification, UCPEstimates, GuidanceForRequirements

asserted axioms:
RequirementsSpecifier $\sqsubseteq$ RoleDefinition $\sqcap$
$\forall$responsible (FunctionalSpecification $\sqcap$ UCPEstimates)
CreateRequirements $\sqsubseteq$ TaskDefinition $\sqcap$
$\forall$mandatoryOuput.FunctionalSpecification $\sqcap$
$\forall$optionalOutput.UCPEstimates $\sqcap$
$\forall$mandatoryInput.Nothing $\sqcap$
$\forall$optionalInput.Nothing …

Figure 26. Excerpt of the requirements-method ontology

Figure 26 illustrates an important property of the ontology specification that is so-called open-world assumption (Horrocks, 2003). It states, intuitively, that anything not explicitly expressed by an ontology is unknown. For example, seeing that we wanted to state that the Create Requirements class has not any mandatory inputs, we stated that the Task Definition "Create Requirements" has mandatory input only Nothing. However, it can be proved that

$$\text{spem} \vdash \text{requirements-method} \qquad (12)$$

But if we state that the Role Definition "Requirements Specifier" is responsible for Guidance "Guidance for Requirements" as follows:

$$\text{RequirementsSpecifier} \sqsubseteq \exists\ \text{responsible.GuidanceForRequirements} \tag{13}$$

then this asserted axiom is in contradiction with the asserted axiom stating that a Role Definition is responsible only for a Work Product Definition, thus the Role Definition “Requirements Specifier” and transitively the Task Definition “Create Requirements” classes become unsatisfiable, thus even the ontologies are inconsistent, therefore it is true that

$$\text{spem} \ |\!/\!-\ \text{requirementsX-method} \tag{14}$$

Figure 27 shows the visualized inferred model of the ontology “requirementsX-method” that contain mentioned contradictory axiom. As the figure illustrates, the reasoning process infers that the Create Requirements class and the Requirements Specifier class are subclasses of the class Nothing, thus they are unsatisfiable.

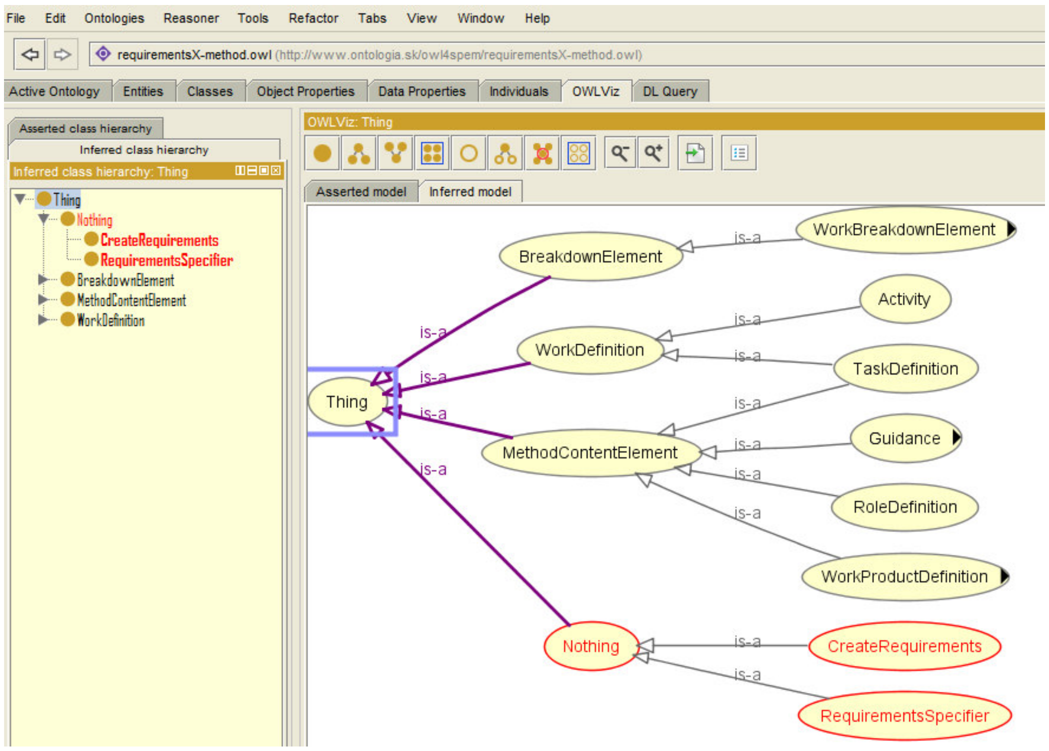


Figure 27. Example of inferred inconsistency between the SPEM Ontology and a SPEM method ontology

The second example presents the second validation scenario that is a SPEM process model validation with ontology. We present also an excerpt of a SPEM process ontology to be more demonstrative about its structure on the one hand, and to present inconsistency more precisely on the other hand. Figure 28 presents main concepts and their relation of the Requirements-process ontology that is also depicted with the SPEM UML Profile in Figure 5.

requirements-process ontology

imports:
spem, requirements-method

owlClasses:
FunctionalSpecification_v1, FunctionalSpecification_v2, UCPEstimates_v1, RequirementsSpecifier_I1, RequirementsSpecifier_I2, CreateRequirements_I1, CreateRequirements_I2

asserted axioms:
FunctionalSpecification_v1 $\sqsubseteq$ WorkProductUse $\sqcap$ 1 contentTrace.FunctionalSpecification
RequirementsSpecifier_I1 $\sqsubseteq$ RoleUse $\sqcap$ 1 contentTrace.RequirementsSpecifier
CreateRequirements_I1 $\sqsubseteq$ TaskUse $\sqcap$ 1 contentTrace.FunctionalSpecification $\sqcap$
$\forall$ mandatoryOutput.FunctionalSpecification_v1 $\sqcap$
$\forall$ optionalOutput.Nothing $\sqcap$
$\forall$ mandatoryInput.Nothing $\sqcap$
$\forall$ optionalInput.Nothing $\sqcap$
$\forall$ performs.RequirementsSpecifier_I1 …

Figure 28. Excerpt of the requirements-process ontology

The first three asserted axioms illustrated in Figure 28 represent method content separation from the process together with the appropriate ContentTrace relation. It can be proved that

$$\text{SPEM Ontology} \vdash \text{requirements-method} \vdash \text{requirements-process} \tag{15}$$

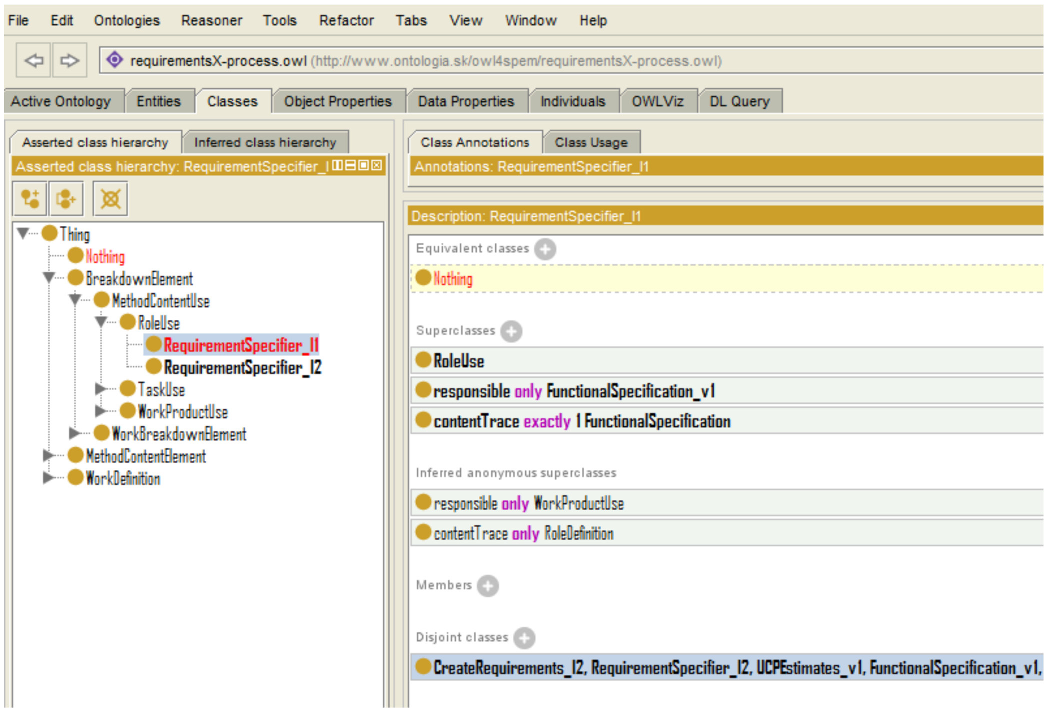
But if we state for example, that the Role Use "Requirements Specifier I1" traces the Work Product "Functional Specification" from the method content ontology "requirements-method" as follows:

$$\text{RequirementSpecifier\_I1} \sqsubseteq 1\ \text{contentTrace.FunctionalSpecification} \tag{16}$$

then it is true that

$$\text{SPEM Ontology} \vdash \text{requirements-method}\ |/\text{-}\ \text{requirementsX-process} \tag{17}$$

For sake of variability Figure 29 does not depict inferred visualized taxonomy of the ontology "requirementsX-process" but rather the details of the Role Use "Requirements Specifier I1". Since we have stated that it traces to the Work Product "Functional Specification" the class became unsatisfied, therefore the ontology "requirementsX-process" is inconsistent.


File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query
Asserted class hierarchy Inferred class hierarchy
Class Annotations Class Usage

requirementsX-process.owl (http://www.ontologia.sk/owl4spem/requirementsX-process.owl)

Asserted class hierarchy: RequirementSpecifier_I1

- Thing
  - Nothing
  - BreakdownElement
    - MethodContentUse
      - RoleUse
        - RequirementSpecifier_I1
        - RequirementSpecifier_I2
      - TaskUse
      - WorkProductUse
    - WorkBreakdownElement
  - MethodContentElement
  - WorkDefinition

Annotations: RequirementSpecifier_I1

Description: RequirementSpecifier_I1

Equivalent classes
Nothing

Superclasses
RoleUse
responsible only FunctionalSpecification_v1
contentTrace exactly 1 FunctionalSpecification

Inferred anonymous superclasses
responsible only WorkProductUse
contentTrace only RoleDefinition

Members

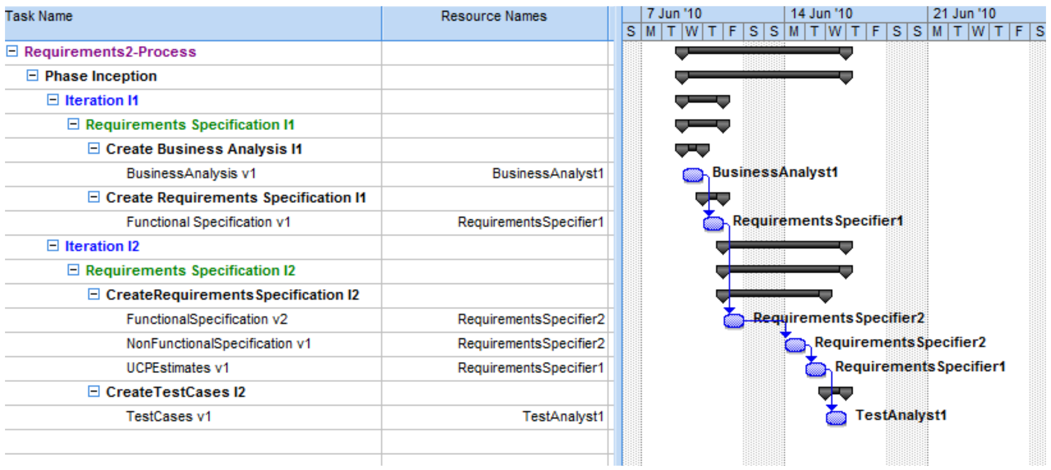
Disjoint classes
CreateRequirements_I2, RequirementSpecifier_I2, UCPEstimates_v1, FunctionalSpecification_v1,

Figure 29. Example of inferred inconsistency between the SPEM Ontology, a SPEM method ontology and a SPEM process ontology

6.2 Project plan verification with ontology

The previous examples presented the implementation of the approach to SPEM model verification with ontology. The validation process was based on OWL DL reasoning with the SPEM Ontology, a SPEM method ontology and a SPEM process ontology. Since Table 4 depicted mapping between these ontologies and the main components of a Knowledge Representation System, only TBox was used in the inference. The follow examples present the second approach developed in this work that is the ontology based approach to project planning. The approach is intended for use in two utilization scenarios, the project plan generation from the ontology and the project plan verification with ontology. Seeing that the former is based on XSL transformation from a SPEM process ontology and consists of utilization scenario that was already presented (a SPEM process model verification with ontology), we present only the latter.

This approach reuses the reasoning between ontologies mentioned in the previous approach and it additionally uses the individuals obtained from a project plan, which are the instances of a SPEM process ontology. Therefore the OWL DL verification uses ABox component either. Figure 30 depicts a project plan that contains two iterations of extended requirements specification methods that we used in previous.



Task Name	Resource Names
⊟ Requirements2-Process	
⊟ Phase Inception	
⊟ Iteration I1	
⊟ Requirements Specification I1	
⊟ Create Business Analysis I1	
BusinessAnalysis v1	BusinessAnalyst1
⊟ Create Requirements Specification I1	
Functional Specification v1	RequirementsSpecifier1
⊟ Iteration I2	
⊟ Requirements Specification I2	
⊟ CreateRequirementsSpecification I2	
FunctionalSpecification v2	RequirementsSpecifier2
NonFunctionalSpecification v1	RequirementsSpecifier2
UCPEstimates v1	RequirementsSpecifier1
⊟ CreateTestCases I2	
TestCases v1	TestAnalyst1

Figure 30. Example of project plan

As the Figure illustrates, the Process "Requirements2-Process" has one Phase "Phase Inception" that consists of two Iterations "Iteration 1" and "Iteration 2". Each iteration consists of the same Activity "Requirements Specification", which is configured differently in each iteration. For example, the Task Use "Create Business Analysis" and "Create Test Cases" are included just in the one iteration, while the Task Use "Create Requirements" is used in both. Moreover, the use of the Task Use "Create Requirements" is different in the first iteration, where just the Work Product Use "Functional Specification v.1" is the output, while the Work Product Uses "Functional Specification v.2", "Nonfunctional Specification v.1" and "UCP Estimates v.1" are the outputs in the second iteration. The excerpts of the formal models of the necessary method content and process ontology can be defined in the same way as it is shown in Figure 26 and 28, where just additional ontology classes have to be added. Therefore we do not depict them as well. However they are attached to this thesis, where requiterements2-method is the SPEM method ontology and requirements2-process is the SPEM process ontology.

Since we want to execute the reasoning process we need to transform the project plan depicted in Figure 30 to ontology. As it was mentioned, the project plan is enacted with the SPEM process with instantiation relation. Figure 31 depicts the formal model of the ontology "projectplan1" that was generated from the project plan depicted in Figure 30.

projectplan1 ontology

imports:
spem, spembaseplugin, requirements2-method, requirements2-process

individuals:
Requirements2-Process, PhaseInception, IterationI1, IterationI2, RequirementsSpecificationI1, FunctionalSpecification_v1, FunctionalSpecification_v2, UCPEstimates_v1, RequirementsSpecifier_I1, RequirementsSpecifier_I2, CreateRequirements_I1, CreateRequirements_I2 …

instantiation:
Process(“Requirements2-Process”), PhaseInception(“PhaseInception”)…
BusinessAnalyst_I1(“Business Analyst1”) …
CreateBusinessAnalysis_I1(“CreateBusinessAnalysis_I1”)…
BusinessAnalysis_I1(“BusinessAnalysis_I1”)

objectproperty assertions:
responsible(“Business Analyst1”,” BusinessAnalysis_I1”)
performs(“CreateBusinessAnalysis_I1”,” Business Analyst1”) …

Figure 31. Example of the project plan ontology

The figure can raise a question why, for example, we have stated the Iteration1 as a class with the instance “Iteration1”? We could use the simplest solution where the class is Iteration with instance “Iteration1”. Even it seems to be simpler and more transparent we have decided to do not use such solution. The main reason is because we wanted to be conformed to the normative specification of a SPEM model, which must be represented with the SPEM UML Profile. The SPEM UML Profile does not provide set of stereotypes for individuals, but only for classes. Since we wanted to create the approach of ontology based SPEM utilization as most usable as possible, we must conform to its definition.

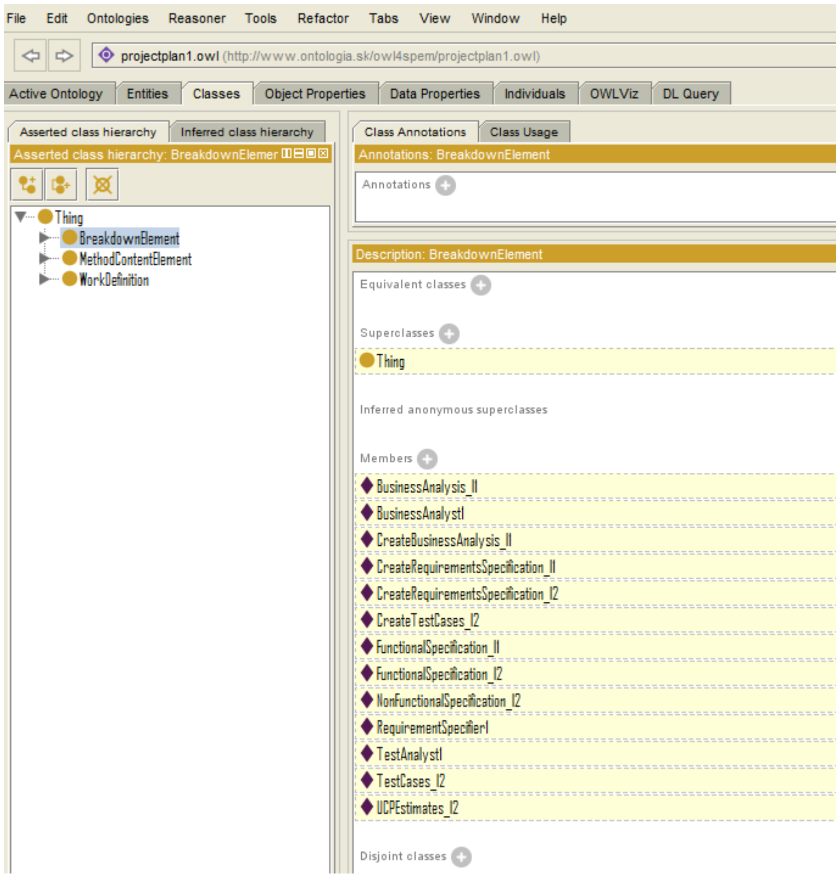
So we define a project planning knowledge as the union of a SPEM method ontology and a SPEM process ontology together with the SPEM Ontology and the SPEM Base Plugin Ontology. Formally:

$$\text{Project Planning Knowledge} = \text{SPEM Ontology} \sqcup \text{SPEM Base Plugin Ontology} \sqcup \text{requirements2-method} \sqcup \text{requirements2-process} \tag{18}$$

It can be proved that

$$\text{projectplan1} \models \text{Project Planning Knowledge} \tag{19}$$

Figure 32 depicts the individuals of the project plan 1 in the ontology “projectplan1” in Protégé. The figure illustrates the result of the realization process which was provided by the Pellet reasoner. Since the all classes of instantiated individuals have the Breakdown Element as a superclass, the realization process inferred that the Breakdown element is their class either.



projectplan1.owl (http://www.ontologia.sk/owl4spem/projectplan1.owl)

Asserted class hierarchy: BreakdownElement

- Thing
  - BreakdownElement
  - MethodContentElement
  - WorkDefinition

Annotations: BreakdownElement

Annotations

Description: BreakdownElement

Equivalent classes

Superclasses

- Thing

Inferred anonymous superclasses

Members

- BusinessAnalysis_I1
- BusinessAnalystI
- CreateBusinessAnalysis_I1
- CreateRequirementsSpecification_I1
- CreateRequirementsSpecification_I2
- CreateTestCases_I2
- FunctionalSpecification_I1
- FunctionalSpecification_I2
- NonFunctionalSpecification_I2
- RequirementSpecifierI
- TestAnalystI
- TestCases_I2
- UCPEstimates_I2

Disjoint classes

Figure 32. Example of project plan individuals realization

If we change in the project plan that the Work Product Use “Nonfunctional Specification” is the mandatory output from the Task Use “Create Business Analysis” as follows:

$$\text{mandatoryOutput(CreateBusinessAnalysis\_I1, NonfunctionalSpecification)} \qquad (20)$$

then it is true that

$$\text{projectplan1X} \;|\!\!\neq \text{Project Planning Knowledge} \qquad (21)$$

6.3 Software project enactment with a supplier

The last example that is presented in this thesis is the ontology based approach to software project enactment with a supplier. We have introduced its four possible utilization scenarios that are: the verification of the set of SPEM methods with ontology, verification of the set of SPEM processes with ontology, project plan generation with the set of method plugins with ontology and the project plan verification with the set of method plugins with ontology. In this example we describe just the scenario of the verification of the set of SPEM methods with ontology, because the others are constituted with the utilization scenarios that were already presented. Uniqueness of this example is based on creation of a necessary mapping between two method contents to execute the OWL DL reasoning.

As we have already mentioned this approach is indented for reasoning with more than one method content or one process. We use the additional SPEM concept that is the Method Plugin. Since a method plugin consists of a method content and a process, we simply reuse already defined method content and process from the previous example that constitute a company process-method plugin. Formally

$$\text{company-methodplugin} = \text{requirements-method} \sqcup \text{requirements-process} \qquad (22)$$

Now, a supplier's method plugin ontology has to be created. Its formal model is depicted in Figure 33.

supplier-methodplugin ontology

imports:
spem, usecase-method

owlClasses:
UsecaseSpecifier, CreateSRS, SRSDocument

asserted axioms:
$\text{CreateSRS} \sqsubseteq \text{WorkProductDefinition} \sqcap \forall \text{mandatoryOutput.SRSDocument} \sqcap \forall \text{performs.UsecaseSpecifier}$
$\text{UsecaseSpecifier} \sqsubseteq \text{RoleDefinition} \sqcap \forall \text{responsible. SRSDocument}$

Figure 33. Supplier's method plugin ontology

The figure illustrates that the supplier's method plugin ontology is use case driven. Now, the objective is to verify the consistency between the company's method plugin and the supplier's method plugin. The necessary condition is to create mapping between all elements of both ontologies. Since the supplier's method plugin consists only of method content elements, the mapping is only between method classes as it is illustrated in Table 5.

Table 5. The mapping between the company's and supplier's method plugin

company's method class	supplier's method class	mapping type
Requirements Specifier	Usecase Specifier	sameClassAs
Create SRS	Create Requirements	sameClassAs
Functional Specification	SRSDocument	subClassOf

We have stated the third mapping with the subClassOf relation, because we want to present the usability of our method. For example, let the supplier uses the Software Requirements Specification (SRS) for functional and also nonfunctional requirements specification. Since the company's Work Product „Functional Specification" contains only functional requirements, the object property "subClassOf" had to be used. Note that mappings can be stated between object properties either, but since the both ontologies are represented with SPEM ontology that is not necessary. Based on previous it can be proved that

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{companyprocess-methodplugin} \vdash \text{supplierprocess-methodplugin}$$

thus both ontologies are consistent. Bu it we modify supplier's method plugin as Formula 23 states, and set the additional condition during mapping process as Formula 24 states

$$\text{CreateSRS} \sqsubseteq \forall \text{mandatoryInput.BusinessVision} \tag{23}$$

$$\text{BusinessVision} \sqcap \text{FunctionalSpecification} = \varnothing \tag{24}$$

then it is true that:

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{companyprocess-methodplugin} \;|/\!\!-\; \text{supplierprocess-methodplugin} \tag{25}$$

thus the ontologies are inconsistent, hence it is not possible to enact software project between the company and its supplier. Figure 34 depicts the result of the reasoning process. As it is illustrated, the Work Product Definition “Functional Specification“ and “SRS Documents“ became unsatisfied classes, therefore both ontologies are inconsistent.

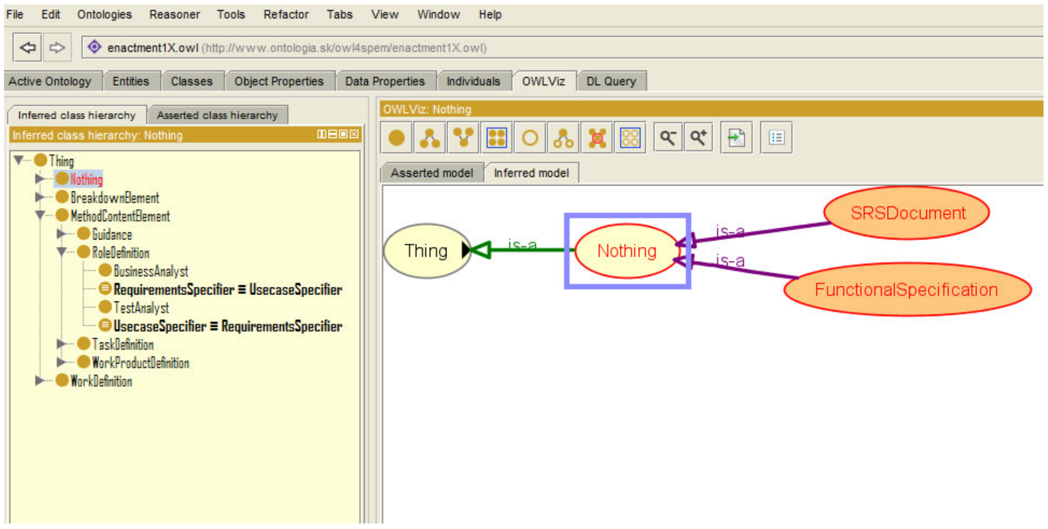

File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
enactment1X.owl (http://www.ontologia.sk/owl4spem/enactment1X.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query
Inferred class hierarchy Asserted class hierarchy
Inferred class hierarchy: Nothing
Thing
Nothing
BreakdownElement
MethodContentElement
Guidance
RoleDefinition
BusinessAnalyst
RequirementsSpecifier ≡ UsecaseSpecifier
TestAnalyst
UsecaseSpecifier ≡ RequirementsSpecifier
TaskDefinition
WorkProductDefinition
WorkDefinition
OWLViz: Nothing
Asserted model Inferred model
Thing
is-a
Nothing
is-a
is-a
SRSDocument
FunctionalSpecification

Figure 34. Example of the inconsistency between two separated method plugins

7 Conclusion

First, this chapter presents the contributions of this thesis. Then we compare them to the state of the art that we have presented in the Chapter 1. Finally we discuss our future work.

7.1 Contributions

7.1.1 Scientific contributions

7.1.1.1 SPEM transformation to the Semantic Web

To our knowledge, there have been no works carried out that analyze how the SPEM metamodel can be transformed to the Semantic Web Technical Space. We have provided the comprehensive study of the transformation of the SPEM metamodel, a SPEM method content model, a SPEM process model and a SPEM method plugin to the OWL DL Ontology. The standard SPEM conceptual framework provides its capabilities with the metamodel approach, thus it is apparent that the ontology based approach improves its capabilities, since the ontology in general clarifies the knowledge structure, reduces conceptual and terminological ambiguity and allows the sharing of knowledge. More precisely, the SPEM architecture based on the SPEM metamodel and the SPEM UML Profile is not capable to make statements about what can be expressed in a SPEM model with the DL based formal system, unlike than our ontology based approach, that is.

7.1.1.2 Ontology based utilizations of the SPEM Ontology

The work has presented three approaches that utilize created ontologies. The approach to SPEM model validation, a project plan verification and a software project enactment with a supplier. Every approach has presented its ontology based architecture with the appropriate mapping; the engines based on the OWL DL reasoning and proposed usage scenarios. Every scenario was supplemented with the formal model that defines conditions to achieve a positive and negative result.

7.1.2 Engineering contributions

7.1.2.1 Usage scenarios implementation

Chapter 6 has presented the implementation of every proposed approach. We have provided the excerpts of the key parts of the used ontologies to present approaches with the example studies. Every example discussed in this chapter was fully tested and is attached to this work. The implementation addresses the usage scenarios that were proposed in the corresponded approach.

7.1.2.2 OWL4SPEM framework

We provide OWL4SPEM framework v.1 that is the set of necessary XSL transformations needed to implement custom usability of proposed scenarios. The *SPEMMethodContent2OWL* XSL transformation allows a SPEM method ontology generation from a SPEM method content model, *SPEMProcess2OWL* allows a SPEM process ontology generation from a SPEM process model and finally the *MPP2OWL* transformation allows a SPEM Process individuals generation from a project plan. The framework was tested with the Enterprise Architect version 7.1.832, the Protégé version 4.0.2 with the Pellet Plugin and MS Project Plan 2007.

7.2 Comparison with the related works

The state of art presented in this thesis was divided to the three major groups. The works that concern with ontologies in software engineering in general constitute the first group. The works that concern with the usability of ontologies in the MDA technical space constitute second group. Finally, the works that are closest to our work, which focus on the utilization of ontologies for SPEM constitute third group. Hence the evaluation of the thesis' contributions is primarily focused against this group. Note that since SPEM is MDA's standard, the third group is subset of the second.

The papers presented in the chapter 1.2.2 were distinguished into two main categories, SET domain ontologies and ontologies as software artifacts (Callero, 2006). Just the former category is similar like the subject of this thesis, since it refers to the ontologies whose main goal is to represent (at least partially) knowledge of a certain subdomain within SET matter. However, we apprehend the works contained in this category as not competitive but rather supplementary. The works (Mendes 2005) or (Sicilia, 2005) propose ontologies of SWEBOK. The objectives of SWEBOK are much broader than objectives of SPEM. For example, the SWEBOK objectives are: to clarify the place and set the boundary of software engineering with respect to other disciplines such as computer science, project management, computer engineering, and mathematics; to characterize the contents of the software engineering discipline; or to provide a topical access to the SWEBOK, which is divided into knowledge areas such as the Software Requirements, the Software Design or the Software Engineering Process knowledge area (IEEE, 2004). The last mentioned consists of the Notations for Process Definition subarea that references SPEM. Hence the SPEM ontology that we have created is rather supplementary then competitive to the SWEBOK ontologies. However the mutual relationship between a SWEBOK ontology and the SPEM ontology should be accomplished and utilized. Since this is a subject of our future research, we discuss it in the corresponding chapter of this work.

The second group presented in chapter 1.2.3 is constituted with works that concern with combining MDA technical space and the Semantic Web technical space. Again, our intention was not to compete to these works, but rather to reuse their contributions as most as possible. In our thesis, we have merged the SPEM metamodel to the SPEM UML Profile hence we were able to establish a mapping between the SPEM UML Profile and UML Ontology Profile presented in Table 3. Thus we have utilized and conformed to the hallmark work (Gašević, 2009) that proposes transformation of a MDA standard to the Semantic Web technical space with a mapping between UML Ontology profile and an arbitrary UML Profile.

The closest papers to our work described in the chapter 1.2.3.1 are works that also intends to use SPEM in the Semantic Web technical space, each for its own purpose. The first work has presented the representation of SPEM in DL; the second has presented the approach to SPEM process constraint definition with the semantic rules using SWRL and the third has presented the approach for generating assessment for the SCRUM software process. When we compare our work with these works, we conclude that we have created

a) the more comprehensive analysis of the SPEM metamodel and the SPEM UML Profile in order to create a SPEM ontology than every mentioned work
b) the more accurate ontology architecture for SPEM with respect to the actual SPEM metamodel than every mentioned work
c) the additional ontology based approach to SPEM model validation with implementation
d) the more accurate ontology based approach to a project plan verification with implementation than the first and third work
e) the additional ontology based approach to software project enactment with a supplier with implementation

Explanation:

- The reason of (a) is because the second and the third work did not analyze the SPEM metamodel with concern of creating an ontology of SPEM as comprehensive as we did. The second paper depicts just an excerpt of the SPEM metamodel and concludes that “the most SPEM terms can be directly translated into OWL”. Third paper does not discuss SPEM metamodel at all. First work has that intention, but unfortunately the paper is outdated and concerns with the SPEM metamodel version 1.1, that did not support the separation of method content from process; and on the contrary, the papers proposes transformation of SPEM metamodel constraints in OCL to the ontology, but the actual SPEM metamodel does not support OCL anymore.
- The (b) is implied from (a).
- None of the works have presented an approach to neither a SPEM method model nor a SPEM process verification with a SPEM ontology, hence the (c) is true.
- Just the second approach has concerned with the project plan verification with a SPEM ontology. Even (a, b, c, e) are true we cannot to say that our approach of project plan verification is more accurate (with respect of SPEM metamodel), rather is more comprehensive. If we focus only on the substantive part of the approach, the second work uses instantiation relation between project plan and a SPEM process either, thus it conforms to the project plan enactment with a SPEM process as it SPEM defines. Moreover, the approach proposes use of SWRL that add rules to the reasoning process. Thus we conclude that the approach proposes reasoning with advanced expressiveness.

- None of the presented works have concerned with the reasoning of more method contents or processes, thus (e) is true either.

It should be noted that this thesis has improved our previously published works, i.e. the approach to SPEM transformation to the Semantic Web technical space (Líška, 2009a), the approach to SPEM model validation (Líška, 2009b), the approach to project plan verification with ontology (Líška, 2010a) and the ontology based approach to software project enactment with a supplier (Líška, 2010c). These were initial works that have proposed that a Method Content Use element is the subclassOf a Method Content Element; hence we were not compliant with the SPEM conceptual framework that separates reusable method content from a process. The presented work in this thesis avoided this inconsistency, because it states that a Method Content Use element traces Method Content Element. Thus in this thesis, the separation of method content form process is already fully supported. In order to this little change, we had to correct implementation either.

7.3 Future work

Even we have presented an ontology based architecture of SPEM that covers its key architectural components such as SPEM Method Content, SPEM Process or SPEM Method Plugin, there is still a lot work that needs to be done in this area. We aimed to continue to the three main directions. The first is focused to the SPEM Ontology improvement, second is to include SWEBOK ontology to the reasoning process and the third concerns with a utilizations improvement.

7.3.1 SPEM Ontology improvement

The solution that we have presented is not complete from the SPEM metamodel point of view. The main focus was to provide utilizations of the SPEM ontology in the context of concrete software engineering areas such as model validation, project plan verification and software project enactment with a supplier. Therefore we did not create the SPEM ontology in the same broadness as the SPEM metamodel is. For example we did not transform the SPEM metamodel concepts such the Variability Element that provides capabilities for content variation and extension to a specific list of SPEM classes. Thus the one major research direction is to complete the SPEM ontology with additional SPEM metamodel semantics which we have avoided. Another important direction is to include SWRL to the SPEM Ontology to provide more expressiveness for the reasoning process.

7.3.2 Include SWEBOK in the reasoning process

Since the SWEBOK contains knowledge about many software engineering knowledge areas, we could utilize it to extend knowledge base for the reasoning processes that we have presented in this work. By other words, we want to implement the presented our solutions to the other areas of software engineering, such as software design, software construction etc. We aim to achieve this goal with including of a content of a Work Product to the reasoning process. For example “does a requirements specification contain uses cases?”, or additionally “does the use cases trace the business processes in business analysis?” etc.

7.3.3 Implementation improvement

We have provide the OWL4SPEM framework that consists of set of XSL transformations that can be used to transform custom SPEM model to the ontology for reasoning purposes. But it is very difficult to imagine that for a purpose of project plan verification a project manager will use ontology editor or a knowledge based framework directly, without appropriate user interfaces. Therefore, we have started implementation of a macro for the MS Project that will remotely access OWL API for OWL-DL reasoning, which will print the verification results back into MS Project Plan with appropriate language.

Acknowledgments. "This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/508/09."

30.5.2010

http://www.fiit.sk

For further information please visit http://www.ontologia.sk/owl4spem/,
or send email to liska@semantickyweb.sk

References

ATKINSON, C., KUHNE, T. 2002. Profiles in a strict meta-modeling framework. *Science of Computer Programming*, 2002, vol. 44, no. 1, pp. 5–22.

BAADER, F., HORROCKS, I., SAATLER, U. 2004. Description Logics. In *Handbook on Ontologies, International Handbooks on Information Systems*, Springer, 2004, pp. 3-28.

BÉZIVIN, J., BUTTNER, F., GOGOLLA, M. et al. 2006. Model transformations? Transformation Models! In *Proceedings of the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems*, Genoa, 2006, Italy, pp. 440–453.

BRICKEY, D., GUHA, R. V., MCBRIDE, B. 2004. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 2004, W3C.

CALERO, C., RUIZ, F., PIATTINI, M. 2006. *Ontologies for Software Engineering and Software Technology*, Springer, 2006, Heidelberg.

CRANEFIELD, S. 2001. Networked Knowledge Representation and Exchange using UML and RDF, *Journal of Digital Information*, 2001, vol. 1, no. 8.

DJURIĆ, D., GAŠEVIĆ, D., DEVEDŽIĆ, V. 2005. Ontology Modeling and MDA. *Journal of Object Technology*, 2004, vol. 4, no. 1, pp. 109-128.

FALBO, R. A., GUIZZARDI, G., DUARTE, K. C. 1992. An Ontological Approach to Domain Engineering. In *Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Ischia, 1992, pp. 351-358.

FAVRE, J. M., NGUYEN, T. 2005. Towards a megamodel to model software evolution through transformations. *Electronic Notes in Theoretical. Computer Science*, 2005, vol. 127, no. 3, pp. 59–74.

FRANKEL, D. S. 2003. *Model Driven Architecture. Applying MDA to Enterprise Computing*, Willey, 2003, USA.

FUJITA, H., ZUALKERNAN, I. A. 2008. An Ontology-Driven Approach for Generating Assessments for the Scrum Software Process. In *Proceedings of the seventh SoMeT_08*. IOS Press 2008, The Netherlands, pp. 190-205.

GAŠEVIĆ, D., DJURIĆ, D., DEVEDŽIĆ, V. 2005. Bridging MDA and OWL Ontologies, *Journal of Web Engineering*, vol. 4, no. 2, pp. 119-134.

GAŠEVIĆ, D., DJURIĆ, D., DEVEDŽIĆ, V. 2009. *Model Driven Engineering and Ontology Development*, 2nd ed., Springer, 2009, Berlin.

GÓMEZ-PÉREZ, A., FERNÁDEZ-LÓPEZ, M., CORCHIO, O. 2004. *Ontological Engineering*, Springer-Verlag, 2004, London.

GRUBER, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition,* vol. 5, no. 2, pp. 199–220.

GUARINO, N. 1995. Formal ontology, conceptual analysis and knowledge representation, *International Journal of Human-Computer Studies*, vol. 43, no. 5/6, pp. 625–640.

HAPPEL, H. J., SEEDORF, S. 2006. Applications of ontologies in software engineering, In *International Workshop on Semantic Web Enabled Software Engineering* (SWESE'06), 2006, Athens.

HART, L., EMERY, P., COLOMB, B., et al. 2004. OWL Full and UML 2.0 Compared, *OMG TFC Report*, 2004, OMG.

HENDLER, J. 2001. Agents and the semantic web. *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 30–37.

HILERA, J. R., SÁNCHEZ-ALONSO, S., GARCÍA, E., et al. 2005. OntoGLOSE: A Light-weight Software Engineering Ontology. *First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE)*, Alcalá de Henares, 2005, Spain.

HORROCKS, I., PATEL-SCHNEIDER, P., F., BOLEY, H., et al. 2004. SWRL: A Semantic Web Rule Language, Combining OWL and RuleML, 2004, *W3C Member Submission*.

HORROCKS, I., PATEL-SCHNEIDER, P., van HARMELEN, F. 2003. From SHIQ and RDF to OWL: The making of a web ontology language, *Journal of Web Semantics*, 2003, vol. 1, no. 1, pp. 7–26.

CHANDRASEKARAN, B., JOSEPHSON, J. R., BENJAMINS, V. 1998. Ontology of Tasks and Methods. In *Proceedings of KAW'98*, Banff, 1998, Canada.

IEEE, 2002. IEEE Std 610.12-1990(R2002). *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, 2002, USA.

IEEE, 2004. *SWEBOK - The Guide to the Software Engineering Body of Knowledge 2004 VERSION*, IEEE Computer Society.

KLEPPE, A., WARMER, J., BAST, W. 2003. *MDA Explained: The Model Driven Architecture: Practice and promise*, Addison-Wesley, 2003, Reading, MA.

KRDŽAVAC, N., GAŠEVIĆ, D., DEVEDŽIĆ, V. 2009. Model Driven Engineering of a Tableau Algorithm for Description Logics. *Computer Science and Information Systems*, 2009, vol. 6, no. 1.

KRUCHTEN, P. 2003. *The Rational Unified Process: An Introduction. (3rd edition)*, Addison-Wesley, 2003, USA.

KURTEV, I., BÉZIVIN, J., AKSIT, M. 2002. Technological spaces: An initial appraisal. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, and ODBASE, Industrial Track*, Irvine, 2002, CA.

KURTEV, I., van den BERG, K. 2003. Model Driven Architecture based XML processing. *Proceedings of the ACM Symposium on Document Engineering*, Grenoble, France, pp. 246–248.

KVASNIČKA, V., POSPÍCHAL, J. 2005. *Mathematic logic*, 2005, Slovak University of Technology in Bratislava.

LARBURU, I. U., PIKATZA, J. M., SOBRADO, F. J., et al. Hacia la implementación de una herramienta de soporte al proceso de desarrollo de software. *Workshop in Artificial Intelligence Applications to Engineering (AIAI)*, San Sebastián, 2003, Spain.

LASSILA, O., MCGUNIESS, D. 2001. The Role of Frame-Based Representation on the Semantic Web. *KSL Technical Report No. KSL-01-02*, 2001.

MCGUINESS, D. L., HARMELEN, F. 2004. OWL Web Ontology Language Overview. *W3C Recommendation*, 2004, W3C.

NÁVRAT, P. et al. 2002. *Artificial Intelligence,* 2002, Slovak University of Technology in Bratislava.

MENDES, O., ABRAN, A. 2005. Issues in the development of an ontology for an emerging engineering discipline. *First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE)*, Alcalá de Henares, 2005, Spain.

OMG. 2006a. *Meta Object Facility (MOF) 2.0 Core Specification*, 2006, OMG, USA.

OMG. 2006b. *Object Constraint Language, v. 2.0.* , 2007, OMG, USA.

OMG. 2007. *MOF 2.0 / XMI Mapping Specification*, 2007, OMG, USA.

OMG. 2008. *Software and Systems Process Engineering Meta-Model 2.0.* 2008, OMG, USA.

OMG, 2009a: *UML 2.2 Superstructure Specification*, 2009, OMG, USA.

OMG, 2009b. *Ontology Definition Meta-Model 1.0*, 2009, OMG, USA.

OMG, 2009c: *UML 2.2 Infrastructure Specification*, 2009, OMG, USA.

PAN, J., HORROCKS, I. 2001. Metamodeling Architecture of Web Ontology Languages. In *Proceedings of the First Semantic Web Working Symposium*, Stanford, 2001, pp. 131-149.

PIDD, M. 2000. *Tools for Thinking - Modeling in Management Science*, Wiley, 2000, New York.

RECTOR, P., DRUMMONG, N., HORRIDGE, M., et al. 2006. Advanced Reasoning with OWL. In *9th International Protégé Conference*, 2006, Stanford.

RODRÍGUEZ, D., SICILIA, M., A. 2009. Defining SPEM 2 Process Constraints with Semantic Rules Using SWRL. In *Proceedings of the Third International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science held in conjunction with CAiSE'09 Conference*, 2009, Amsterdam, The Netherlands, pp. 95-104.

SHVAIKO, P., EUZENAT, J. 2007. *Ontology Matching*, Springer, 2007, Berlin, Heidelberg.

SEIDEWITZ, E. 2003. What models mean. *IEEE Software*, 2003, vol. 20, no. 5, pp. 26–32.

SICILIA, M. A., CUADRADO, J. J., GARCÍA, E., et al. 2005. The evaluation of ontological representation of the SWEBOK as a revision tool. In *29th Annual International Computer Software and Application Conference (COMPSAC)*, 2005, Edinburgh, UK.

SIRIN, E., PARSIA, B., GRAU, B. C., et al. 2007. Pellet: A practical OWL-DL reasoner, *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 2007, vol. 5, no. 2, pp. 51-53.

SMITH, M. K., WELTY, Ch., MCGUINESS, D. L. 2004. OWL Web Ontology Language Guide, *W3C Recommendation*, W3C, 2004.

SOWA, J. F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole, Pacific Grove, CA.

STEINBERG, D., BUDINSKY, F., PATERNOSTRO, M. et al. 2009. *EMF: Eclipse Modeling Framework (2nd Edition)*, Addison-Wesley Longman 2009, Amsterdam.

SCHWABER, K., BEEDLE, M. 2002. *Agile Software Development with SCRUM*, 2002 Prentice Hall.

USCHOLD, M., GRUNINGER, M. 1996. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 1996, vol. 11(2): pp. 93–15.

VIANU, V. 1997. Rule-based languages. *Annals of Mathematics and Artificial Intelligence*, 1997, vol. 19, no. 1–2, pp. 215–259.

WANG, S., JIN, L., J. CH. 2006. Represent Software Process Engineering Metamodel in Description Logic. In *Proceedings of World Academy of Science, Engineering and Technology*, 2006, vol. 11.

ZUALKERNAN, I. A. 2008. An Ontology-Driven Approach for Generating Assessments for the SCRUM Process. *New Trends in Software Methodologies, Tools and Techniques*. IOS Press, 2008.

Appendix A: Author's publications

LÍŠKA, M., NÁVRAT, P. 2010c. An ontology based approach to software project enactment with a supplier. In *14th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, to be published in *Lecture Notes in Computer Science*, 2010, Springer.

LÍŠKA, M. 2010b. An Approach to Ontology Oriented Employment of SPEM. In *Informatics and Information Technologies Student Research Conference 2010 (IIT.SRC2010)*, FIIT STU, 2010, Bratislava.

LÍŠKA, M., NÁVRAT, P. 2010a. An Approach to Project Planning Employing Software and Systems Engineering Meta-Model Represented by an Ontology, *Computer Science and Information Systems Journal*. (conditional acceptance with minor revision)

LÍŠKA, M. 2009b. An Approach of Ontology Oriented SPEM Models Validation. In *Proceedings of the First International Workshop on Future Trends of Model-Driven Development* in the context of *the 11th International Conference on Enterprise Information Systems*, Milan, Italy, INSTICC Press, 2009, pp. 40-43.

LÍŠKA, M. 2009a. Developing an Ontology for SPEM Method Content Models Validation. In *Informatics and Information Technologies Student Research Conference (IIT.SRC 2009)*, FIIT STU, 2009, Bratislava.

LÍŠKA, M. 2008. Reducing risk of UML model design with axiomatic knowledge architecture. In *ITAT2008*, 2008, Hrebienok.

LÍŠKA, M. 2008. UML model validation with axiomatic knowledge architecture in context of expert system for MDA. In *IWCIT2008-The Seventh International PhD Students' Workshop Control and Information Technology*, 2008, Gliwice, Poland.

LÍŠKA, M. 2007. Constructing expert system for Model Driven Development. In *Proceedings of the Ninth International Conference on Informatics (Informatics'07)*, Slovak Society for Applied Cybernetics and Informatics, 2007, Bratislava.

LÍŠKA, M. 2007. Intelligent automated testing opportunity in Model Driven Development. In *Informatics and Information Technologies Student Research Conference (IIT.SRC 2007)*, FIIT STU, 2007, Bratislava.

LÍŠKA, M. 2007. Model driven development enhancement with analysis agent system. In *AIESA 2007*, FHI EUBA, 2007, Bratislava.

LÍŠKA, M. 2006. Formal Unified Process, In *Informatics and Information Technologies Student Research Conference (IIT.SRC 2006)*, FIIT STU, 2006, Bratislava.

LÍŠKA, M. 2006. Constructing multi-theories expert system for UML models validation. In *1st Workshop on Intelligent and Knowledge oriented Technologies, Slovak Academy of Science*, 2006, Bratislava.

LÍŠKA, M. 2005. Extensional and intensional semantics of PUML objects. In *Informatics and Information Technologies Student Research Conference (IIT.SRC 2005)*, FIIT STU, 2005, Bratislava.

Appendix B: List of attached files

Table 6. List of attached files

File	Type
SPEMMethodContent2OWL.xsl	OWL4SPEM framework v.1
SPEMProcess2OWL.xsl	OWL4SPEM framework v.1
MPP2OWL.xsl	OWL4SPEM framework v.1
spem.owl	OWL4SPEM framework v.1
spembaseplugin-methodplugin.owl	OWL4SPEM framework v.1
requirements-method.owl	example
requirementsX-method.owl	example
requirements-process.owl	example
requirementsX-method.owl	example
requirements2-method.owl	example
requirements2-process.owl	example
projectplan1.mpp	example
projectplan1.owl	example
projectplan1X.mpp	example
projectplan1X.owl	example
companyprocess-methodplugin.owl	example
supplierprocess-methodplugin.owl	example
enactment1.owl	example
supplier2process-methodplugin.owl	example
enactment1X.owl	example