DOI:10.2298/CSIS101230011L

SPEM Ontology as the Semantic Notation for Method and Process Definition in the Context of SWEBOK

Miroslav Líška[1] and Pavol Navrat[1]

[1]Faculty of Informatics and Information Technologies,
Slovak University of Technology,
Ilkovičova 3, 842 16 Bratislava, Slovakia
liska@semantickyweb.sk, navrat@fiit.stuba.sk

Abstract. The Guide to the Software Engineering Body of Knowledge (SWEBOK) provides a consensually validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline. The topic "Notation for Process Definition" references selected notations appropriate for software process definition. However all of them have weakly defined semantics, thus is not possible to use formal techniques for process model creation, validation etc. In this work we present created Software and Systems Process Engineering Meta-Model (SPEM) Ontology that improves the lack of mentioned process notations. The SPEM Ontology constitutes a semantic notation that provides concepts for knowledge based software process engineering. The work also discusses utilization of such semantic notation in other selected SWEBOK topics, the Software Project Planning, the Software Project Enactment, and the Verification and Validation.

Keywords: software and systems process engineering meta-model, web ontology language, model driven architecture, semantic web, SPEM, OWL, MDA, SWEBOK.

1. Introduction

There are a number of notations that are used to define software processes [1]. A key difference between them is in the type of information they define, capture, and use. The approaches encompass for example: natural language [2], Data Flow diagrams [3], Statecharts [4], ETVX [5], Actor-Dependency modeling [6], SADT notation and many others [7]. Unfortunately, semantics of the mentioned notations are defined weakly, thus it is not possible to make and to verify created language statements with formal techniques such as the consistency or satisfiability verification. Although standard software development process frameworks provide much useful information, typically in the form of navigable websites, this information contains only human-readable descriptions. Therefore, these kinds of frameworks cannot be used to represent machine interpretable content [8]. Moreover, these process frameworks are used in the technical spaces [9] that have model based architecture, such as MDA or Eclipse Modeling Framework (EMF) [10]. These kinds of technical spaces also limit knowledge based processing, owing to their weakly defined semantics [11]. However, at present the emerging field of Semantic Web technologies promises new stimulus for Software Engineering research [12]. The acquired opportunity to work with semantics opens door for original contributions to many problems in the field, e.g web service composition aided by semantics [51-53]. 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 [13]. The today's key Semantic Web technology is Web Ontology Language (OWL). OWL 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 [14].

Aforementioned problems in software engineering and facts about the Semantic Web implies an opportunity to support software process definition with OWL, and thus to support software process engineering with knowledge based techniques. In this work we address such an opportunity and propose an ontology based software process definition that could empower software process engineering with knowledge engineering techniques. To achieve it we need to move software process engineering to the Semantic Web technical space. We have chosen Software and Systems Engineering Meta-Model and transformed it to the OWL DL representation, so having created SPEM Ontology.

1.1. Related works

SPEM is MDA standard used to define software and systems development processes and their components [15]. A SPEM process can be systematically mapped to a project plan by instantiating the different process' breakdown structure views. Therefore a SPEM model can represent a knowledge base that can be used for verification, whether a project plan conforms to this knowledge. However, the SPEM metamodel has the semiformal architecture, thus it is not possible to make and to verify created SPEM language statements with formal techniques such as the consistency or satisfiability verification [16]. But if we transform SPEM to the Semantic Web technical space, we can use the mentioned formal techniques due to facilities of OWL. Because SPEM is based on MDA, we can utilize the research results of transforming other MDA's standards to the Semantic Web technical space.

SPEM is specified in the Meta Object Facility (MOF) language 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 [17]. It provides metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems

[18]. 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 Resource Description Framework Schema (RDFS) [19]. 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`, the 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 [20]. MOF is also used for specification of the Unified Modeling Language (UML) that is a language for specification, realization and documentation of software systems [21]. Even if UML and RDFS are similar in the 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 [22]. This difference has also been propagated between OWL and UML [23]. It should be noted that efforts to transfer explicit knowledge into machine processable form encompass a much wider spectrum of works, e.g. [24, 25]. Still others attempt to develop domain specific languages, incorporating knowledge on the domain, that would be adaptable [26] improving in such a way the process of software evolution [27]. At present the main bridge that connects the Semantic Web with MDA is stated in the Ontology Definition Meta-Model (ODM) [28]. 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 mappings between the UML Profile for OWL and other UML Profiles for custom domains [29, 30]. We have already utilized this principle in our previous works where we created an approach to SPEM model validation with ontology [31], an approach to project planning employing software and systems engineering meta-model represented by an ontology [32], and ontology driven approach to software project enactment with a supplier [33]. However, our works are not the only one that concern with using of SPEM in the Semantic Web technical space. In the following paragraph we reference to the three other related works.

The first work proposes to represent SPEM in Description Logic (DL) [34]. The work creates mapping from MOF to DL and mapping from OCL [35] 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 second work presents a competency framework for software process understanding [36]. 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 software process [37] with EPF Composer. However, the third work is the closest to our approach, since it proposes project plan verification with ontology. The work intends to use SPEM process constraint definitions with the semantic rules with Semantic Web Rule Language (SWRL) [38], where SWRL is W3C language that combines OWL and RuleML [39].

1.2. Aims and objectives

We aimed in our research to devise a method that uses ontology based software process notation which could be used for ontology based software process engineering. To be more precise, we propose an extension of the SWEBOK topic “Notation for Process Definition” of the Software Engineering Process Knowledge Area with additional process notation SPEM Ontology. Consequently we present utilizations of such semantic notation in the context of SWEBOK. The SPEM Ontology is first applied to the Software Project Planning topic and then to the Software Project Enactment topics, bought belong to the Software Engineering Management Knowledge Area. Third the SPEM Ontology is discussed in the context of the SWEBOK topic “Validation and Verification“ from the Software Quality Management Process Knowledge Area. For the sake of clarity the utilizations are presented with several usage scenarios defined with description logic.

The rest of the paper is structured as follows. Section 2 presents a method of developing SPEM Ontology based on a transformation from MDA to the Semantic Web technical space. Section 3 presents relationship between SPEM Ontology and SWEBOK. Finally, Section 4 provides conclusion and future research direction.

2. Developing SPEM Ontology

SPEM is MDA standard used to define software and systems development processes and their components [15]. SPEM metamodel is based on MOF and reuses UML 2 Infrastructure Library [40]. Its own extended elements are structured into seven main meta-model packages. SPEM defines three compliance points (CP) above these packages, i.e.: the SPEM Complete CP, the SPEM Process with Behavior and Content CP and the SPEM Method Content CP. The scope of our solution is covered with Compliance Point "SPEM Process with Behavior and Content". The reason of focusing at this compliance point is because we need to work with separated reusable core method content from its application in processes, since a software method content can be used with arbitrary software process, such as iterative, agile etc...

2.1. SPEM conceptual framework

The Software and Systems Process Engineering Meta-model (SPEM) is a process engineering meta-model as well as conceptual framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes [15].

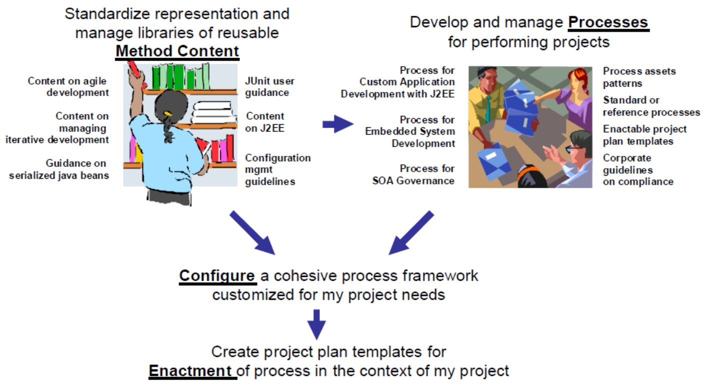
Standardize representation and manage libraries of reusable **Method Content**
Develop and manage **Processes** for performing projects
Content on agile development
Content on managing iterative development
Guidance on serialized java beans
JUnit user guidance
Content on J2EE
Configuration mgmt guidelines
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

Fig. 1. SPEM 2.0's conceptual usage framework

Technically, the separation is represented by SPEM metamodel packages, i.e. the Method Content and the Process with Method metamodel packages. The former provides concepts for SPEM users and organizations to build up a development knowledge base that is independent of any specific processes and development projects. These concepts are the core elements of every method such as Roles, Tasks, and Work Product Definitions etc. The latter necessary metamodel package 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. The most important elements of the Process with Method metamodel package are the Method Content Use elements. These elements are the key concept for realizing the separation of processes from method content and are great capabilities of SPEM. A Method Content Use can be characterized as a reference object for one particular Method Content Element, which has its own relationships and properties. When a Method Content Use is created, it shall be provided with congruent copies of the relationships defined for the referenced content element [15].

The last important metamodel package from the SPEM conceptual framework point of view 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 such that it can be maintained as separate units worked on by distributed teams [15]. 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 for example not every software development process needs to have iterations. Therefore we had to include even the built-in SPEM Base Plugin to our method. It provides commonly used concepts for the domain of software engineering such as Phase, Iteration, Checklist etc.

2.2. Moving SPEM into the Semantic Web

In order to enable use of SPEM in the Semantic Web technical space, we make use of the fact that OWL, ODM and SPEM are serialized in XML format [41]. The OWL Metamodel is a MOF2 compliant metamodel that allows a user to specify ontologies using the terminology and underlying model theoretic semantics of OWL. The mapping between OWL and ODM is expressed in ODM that contains OWL Metamodel [42]. Thus only a mapping between SPEM and OWL is to be created. Since the hallmark work [11] 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 have also used this principle. We have created a mapping between the Ontology UML Profile and the SPEM UML Profile. However, the mapping was not sufficient to create the SPEM Ontology. The main problem was that the SPEM UML Profile does not contain SPEM semantics, and moreover, it was not possible to derive a domain and range of a relationship, etc. Therefore we had decided to create semiautomatic transformation that is based on the merged SPEM metamodel to the SPEM UML Profile, where the result is the SPEM OWL DL Ontology. We have used OWL-DL, because this dialect of OWL retains computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time) [13]. To be conformed to this dialect, we have to adhere that an individual cannot be also a class, what is not violated in MDA technical space because of its 4 meta-layer architecture. For example, an analyst “Slávko Líška” is an instance of a Software Analyst SPEM class that is an instance of the Role Definition SPEM metaclass at the same time, thus the Software Analyst class is an individual and also a class. To avoid this problem in the Semantic Web technical space we have stated that a method content owl class is subclass of a SPEM owl class, and concrete individual is its instance. For example, the individual “Slávko Líška” is the instance of the Software Analyst owl class that is subclass of the Role Definition owl class from the SPEM Ontology. For more detailed and comprehensive description about the SPEM transformation to the Semantic Web technical space and its utilizations, a reader may refer to [43, 44].

3. SPEM Ontology application in the context of SWEBOK

Once we have SPEM Ontology created, we can apply it in various software process engineering cases. Since SWEBOK provides a consensually validated characterization of the bounds of the software engineering discipline and to provide a topical access to the defined software engineering Knowledge Areas [7], we also present the SPEM Ontology utilizations in such manner. The following subsections present SPEM Ontology application in selected topics of the SWEBOK Knowledge Areas.

3.1. Notation for process definition

The first SWEBOK topic that is directly related to the SPEM Ontology is the topic “Notations for Process Definition” of the Software Engineering Process Knowledge Area. Since at present the topic refers to non semantic notations only, the SPEM Ontology introduces new type of notation for process definition that is ontology based. Moreover, SPEM provides concepts also for a method definition that is another added value of such semantic notation. Forasmuch it is necessary to create correct SPEM models (method and process) it is efficient to use automated techniques for the models validation. Seeing that an OWL DL ontology supports reasoning we can also utilize it in a SPEM model verification. Two following scenarios present a SPEM model validation for the SPEM semantics verification.

- 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. 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. Certainly, this validation scenario can be used similarly as 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 only.

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

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

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

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

3.2. Software 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 [45]. Project management is accomplished through the use of processes such as: initiating, planning, executing, controlling and closing [46]. 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 [7]. To support these general objectives, we present an ontology based approach to project planning. We discuss two additional scenarios that could support the ontology oriented software process engineering. So, the third scenario presented in this work is Project plan creation with ontology scenario and the fourth one is Project plan verification with ontology.

- **Scenario 3. Project plan generation with ontology**. When a project manager want to create a project plan, he can create a SPEM method and process models first and then use OWL DL consistency reasoning to ensure that they are consistent. Then he can just simply transform his SPEM process model to the SPEM process ontology.

- **Scenario 4. Project plan verification with ontology**. This scenario is essential when a project manager wants to ensure that his already created project plan is consistent with desired method content and process. However, the scenario usually follows the previous one. A project manager obviously makes many changes to his project plan; therefore it is necessary to ensure that these changes do not violate the required consistency.

Since the third scenario is included in the fourth, we focus only at the Scenario 4. First we define the Project Plan Knowledge as a 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.

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

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

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

From the First Order Logic point of view, the Project Plan Knowledge is the theory and a project plan is its model. Since a theory can have a model only if a theory is consistent [38], it is necessary, that the Formula 5 is either true

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

For more information that includes either example a reader may refer to [32].

3.3. Software project enactment

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 [7]. The ideal state is that a company and its supplier use the same software framework and they use it in the same way. Otherwise risk of budget and time overrun together with quality decrease is greatly increased. Unfortunately, such an ideal state cannot exist. Either companies use different software frameworks, or they use the same software framework - but highly likely in different ways. It is natural that companies have different knowledge acquired from their various projects, and also have different experts with different experiences. Thus even if they use the same software framework, e.g. RUP [45], project enactment with the supplier, due to mentioned differences, is problematic.

Our approach to software process enactment with a supplier is based on OWL DL verification with a set of different method plugins, which represent different methods and processes of a company and its supplier. When OWL

DL verification results in inconsistency, it implies that the 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. Next, there are presented several utilization scenarios of our approach, which extend the overall set of utilization scenarios mentioned in this work.

- 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 those 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 to the previous, but this time processes of a software development are subject for 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 executed in at least two iterations to increase the quality of requirements, but for the supplier,one iteration is also permissible. Again, this is an 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 on at least 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 process from the set of method plugins and 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 is quaranted that the resulted project plan is consistent with desired method plugins.

- Scenario 8 – Project plan verification of 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 as the project plan verification with ontology scenario (i.e., scenario #4). The only difference is in number 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 formally defined conditions that cover the mentioned utilization scenarios. Since the scenario 7 consists of the scenarios 5 and 6 we only present the formal specification of scenarios 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:

$$\text{SPEM method plugin 1 ontology} = \text{SPEM method ontology 1} \cup \text{SPEM process ontology 1} \quad (\mathbf{6})$$

$$\text{SPEM method plugin 2 ontology} = \text{SPEM method ontology 2} \cup \text{SPEM process ontology 2} \quad (\mathbf{7})$$

$$\text{Project Planning Knowledge} = \text{SPEM Ontology} \cup \text{SPEM Base Plugin Ontology} \cup \text{SPEM method plugin 1 ontology} \cup \text{SPEM method plugin 2 ontology} \quad (\mathbf{8})$$

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

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{SPEM method ontology 1} \vdash \text{SPEM method ontology 2} \quad (\mathbf{9})$$

and two SPEM processes are consistent if:

$$\text{SPEM method ontology} \vdash \text{SPEM Base Plugin Ontology} \vdash \text{SPEM method ontology 1} \vdash \text{SPEM method ontology 2} \vdash \text{SPEM process ontology 1} \vdash \text{SPEM process ontology 2} \quad (\mathbf{10})$$

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

$$\text{Project Plan} \models \text{Project Planning Knowledge} \quad (\mathbf{11})$$

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 also Formula 10 must be true. For more informations which include also examples a reader may refer to [32].

3.4. Verification and Validation

The SPEM ontology can also be used in the context of Verification and Validation topic of the Software Quality Management Process Knowledge Area. Verification is an attempt to ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities. Validation is an attempt to ensure that the right product is built, that is, the product fulfills its specific intended purpose [7]. Since SPEM does not concern with the content of work products (e.g. business processes, use cases), is it not possible to verify traceability between these inner elements. Therefore SPEM ontology alone is not sufficient for validation a work product. On the other hand, since the SPEM ontology consists of method and process models, it is possible to use it for evaluating whether a product is build correctly, i.e. with proper method and process. Therefore the SPEM Ontology can provide concepts for a work product verification. Usage scenarios for the Verification and Validation topic are the same as were mentioned in previous subsections.

4. Implementation

Ontologies rely on well-defined and semantically powerful concepts in artificial intelligence [47], such as description logics, reasoning, and rule-based systems [48]. Since we use OWL DL form of ontology, the implementation has a 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 [49]. 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 [11]. As we have it discussed in Subsection 2.2, our approach is conformed to the OWL DL dialect that disallows an individual to be simultaneously a class. Therefore all classes of the ontologies used in our approach constitute TBOX, whereas only individuals obtained from a project plan create ABOX as it is depicted in Table 1.

Table 1. 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
Individuals of a SPEM process ontology	ABox

For the sake of usability we have created OWL4SPEM: a semantic framework for software process engineering. It contain the SPEM Ontology, SPEM Method Plugin ontology, all mentioned XSL transformations that allows generating desired ontologies and lot of examples. For more information a reader may refer to [50].

5. Conclusion

We presented an approach to software process definition with the SPEM Ontology. For the sake of adoptability we presented applications of such semantic notation in the context of selected SWEBOK topics. First we discussed extension of the topic “Notations for Process Definition” with the SPEM Ontology and consequently we presented the ontology in the context of “Software Project Planning”, “Software Project Enactment”, and “Validation and Verification” SWEBOK topics. Since the relationship between the SPEM Ontology and the SWEBOK topics is more comprehensive than we described, it is necessary that the research will continue. However when we compare our approach with work that is perhaps closest to ours [38] it should be noted that we created not only wider method specification, but we also presented its implementation. It supports key property of SPEM, i.e. the Method Content separation from a Process and also the separation from the SPEM Base Plugin. Additionally, since a Method Plugin consists of a Method Content and a Process, our approach can be easily extended with any Method Plugin, for example, with the Rational Unified Process Plugin. However, we are aware that our research must continue in order to be applied successfully in real commercial projects. It is very difficult to imagine that for the purpose of project plan verification a project manager will use 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 purposes and it will print verification results back into MS Project Plan. Additionally, we started to implement a semantic enterprise server with Jena, where the SPEM Ontology will stand as a facility plugin into the architecture. Finally, similarily to other related works, we have to include also SWRL to our approach to extend the expressiveness of description logic with the rule based expressions. The mentioned enhancements to our method are to be viewed as objectives of future research.

Acknowledgments. "This work was supported by the Scientific Grant Agency of Slovakia, grant No. VG1/0508/09, and it is a partial result of the Research & Development Operational Program for the project Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 25240120029, co-funded by ERDF."

References

1. Software Productivity Consortium, "Process Definition and Modeling Guidebook," Software Productivity Consortium, SPC-92041-CMC, (1992)
2. IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology-/Software Life Cycle Processes, vol. IEEE, (1996)
3. ISO/IEC TR 15504:1998, Information Technology - Software Process Assessment (parts 1-9): ISO and IEC, (1998)
4. D. Harel and M. Politi, Modeling Reactive Systems with Statecharts: The Statemate Approach: McGraw-Hill, (1998)
5. R. Radice, N. Roth, A. O. H. Jr. and W. Ciarfella, "A Programming Process Architecture," IBM Systems Journal, vol. 24, iss. 2, 79-90, (1985)
6. E. Yu and J. Mylopolous, "Understanding 'Why' in Software Process Modeling, Analysis, and Design," presented at Proceedings of the 16th International Conference on Software Engineering, (1994)
7. IEEE Computer Society. Software Engineering Body of Knowledge (SWEBOK). Angela Burgess, EUA, (2004)
8. Fujita, H., Zualkernan, I. A. (ed.): An Ontology-Driven Approach for Generating Assessments for the Scrum Software Process. In Proceedings of the seventh SoMeT_08. IOS Press, The Netherlands, 190-205, (2008)
9. Kurtev, I., Bézivin, J., Aksit, M.:Technological spaces: An initial appraisal. In Proceedings of the Confederated International Conferences, CoopIS, DOA, and ODBASE, Industrial Track, Irvine, CA, USA, (2002)
10. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.:EMF: Eclipse Modeling Framework (2nd Edition). Addison-Wesley Longman, Amsterdam, (2009)
11. Gašević, D., Djurić, D., Devedžić, V.: Model Driven Engineering and Ontology Development, 2nd ed., Springer, Berlin, (2009)
12. Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: International Workshop on Semantic Web Enabled Software Engineering (SWESE'06), Athens, USA, (2006)
13. Mcguinness, D. L., Harmelen, F.: OWL Web Ontology Language Overview, W3C Recommendation, (2004). [Online]. Available: http://www.w3.org/TR/owl-features/ (current December 2010)
14. Smith, M.K., Welty, Ch., McGuinness, D.L.: OWL Web Ontology Language Guide, W3C Recommendation, (2004). [Online]. Available: http://www.w3.org/TR/owl-guide/ (current December 2010)
15. Object Management Group: Software and Systems Process Engineering Meta-Model 2.0, formal/2008-04-01. Object Management Group, USA, (2008). [Online]. Available: http://www.omg.org/technology/documents/formal/spem.htm (current December 2010)
16. Krdžavac, N., Gašević, D., Devedžić, V.: Model Driven Engineering of a Tableau Algorithm for Description Logics. Computer Science and Information Systems, Vol. 6, No. 1, (2009)
17. Frankel, D.S.: Model Driven Architecture. Applying MDA to Enterprise Computing. Willey, USA, (2003)
18. Object Management Group: Meta Object Facility (MOF) 2.0 Core Specification, formal/2006-01-01. Object Management Group, USA, (2008). [Online]. Available: http://www.omg.org/spec/MOF/2.0/ (current December 2010)
19. Brickley, D., Guha, R. V., McBride, B.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, (2004). [Online]. Available: http://www.w3.org/TR/rdf-schema/ (current December 2010)

20. Pan, J., Horrocks, I.: Metamodeling Architecture of Web Ontology Languages, In Proceedings of the First Semantic Web Working Symposium, Stanford, 131-149, (2001)
21. Object Management Group: UML 2.2 Superstructure Specification, formal/09-02-03. Object Management Group, USA, (2009). [Online]. Available: http://www.omg.org/technology/documents/formal/uml.htm (current December 2010)
22. Cranefield, S.: Networked Knowledge Representation and Exchange using UML and RDF. Journal of Digital Information, Volume 1 Issue 8, (2001)
23. Hart, L., Emery, P., Colomb, B., Raymond, K., Taraporewalla, S., Chang, D., Ye, Y., Kendall, E., Dutra, M.: OWL Full and UML 2.0 Compared. OMG TFC Report, (2004)
24. Polášek, I., Kelemen, J.: Ontologies in Knowledge Office Systems. In: KEOD 2009, 1st International Conference on Knowledge Engineering and Ontology Development, Funchal - Madeira, Portugal. INSTICC PRESS, 400-413, (2009)
25. Polášek, I., Chudá, D., Kristová, G.: Modelling System Dynamics in a Newer Version of UML (in Slovak). In: Proc. Systémová integrácia 2006. Žilina University, Žilina,311-317, (2006)
26. Hrnčič, D., Mernik, M., Forgáč, M., Kollár, J.: Evolution and Adaptation of Domain Specific Languages. In: Proc. of the Tenth International Conference on Informatics 2009, Technical University of Kosice, Kosice, 154-159, (2009)
27. Kollár, J., Porubän, J., Václavík, P., Bandáková, J., Forgáč, M.: Adaptive Language Approach to Software Systems Evolution. In: Proc. International Multiconference on Computer Science and Information Technology: 1st Workshop on Advances in Programming Languages (WAPL'07), Polish Information Processing Society, 1081-1091, (2007)
28. Object Management Group: Ontology Definition Meta-Model 1.0. formal/2009-05-01. Object Management Group, USA, (2009). [Online]. Available: http://www.omg.org/spec/ODM/1.0/ (current December 2010)
29. Gašević, D., Djurić, D., Devedžić, V.: MDA and Ontology Development. Springer, Berlin, Heidelberg, (2006)
30. Gašević, D., Djurić, D., Devedžić, V.: Bridging MDA and OWL Ontologies. Journal of Web Engineering, Vol. 4, no. 2, pp. 119–134, (2005)
31. Líška, M.: An Approach of Ontology Oriented SPEM Models Validation. In Proceedings of the First International Workshop on Future Trends of Model-Driven Development (FTMDD) in the context of the 11th International Conference on Enterprise Information Systems. Milan, Italy, 40-43, (2009)
32. Líška, M., Návrat, P.: An Approach to Project Planning Employing Software and Systems Engineering Meta-Model Represented by an Ontology. Computer Science and Information Systems Journal (COMSIS), Volume 7, Number 4, 2010, 721-736.
33. Líška, M., Návrat, P.: An Ontology Based Approach to Software Project Enactment with a Supplier. In 14th East-European Conference on Advances in Databases and Information Systems (ADBIS2010), Lecture Notes in Computer Science 6295, Novi Sad, Serbia, Springer, pp. 378-391, (2010)
34. Wang, S., Jin, L. J. CH. Represent Software Process Engineering Metamodel in Description Logic. In Proceedings of World Academy of Science, Engineering and Technology, vol. 11, (2006)
35. Object Management Group: Object Constraint Language 2.2. formal/2010-02-01. Object Management Group, USA, (2010). [Online]. Available: http://www.omg.org/spec/OCL/2.2/ (current December 2010)

36. Zualkernan, I. A. An Ontology-Driven Approach for Generating Assessments for the SCRUM Process. New Trends in Software Methodologies, Tools and Techniques. IOS Press, (2008)
37. Schwaber, K., Beedle, M. Agile Software Development with SCRUM. Prentice Hall, (2002)
38. Rodríguez, D., Sicilia, M., A.: 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. Amsterdam, The Netherlands, pp. 95-104, (2009)
39. Horrocks, I., Patel-Schneider, P., F., Boley, H., Tabet, T., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language, Combining OWL and RuleML. W3C Member Submission, (2004). [Online]. Available: http://www.w3.org/Submission/SWRL/ (current December 2010)
40. Object Management Group: UML 2.2 Infrastructure Specification, formal/2009-02-04. Object Management Group, USA, (2009). [Online]. Available: http://www.omg.org/spec/UML/2.2/ (current December 2010)
41. Object Management Group: MOF 2.0 / XMI Mapping Specification, v2.1.1, formal/2007-12-01. Object Management Group, USA, (2007). [Online]. Available: http://www.omg.org/spec/XMI/2.1.1/ (current December 2010)
42. Djurić, D.: MDA-based ontology infrastructure, Computer Science and Information Systems, Vol. 1, no. 1, pp. 91–116, (2006)
43. Líška, M.: Extending and Utilizing the Software and Systems Process Engineering Metamodel with Ontology. PhD Thesis, ID: FIIT-3094-4984. Slovak University of Technology in Bratislava, (2010)
44. Líška, M. Extending and Utilizing the Software and Systems Process Engineering Metamodel with Ontology. Information Sciences and Technologies, Bulletin of the ACM Slovakia, Vol. 2, No. 2, pp. 8-15, (2010)
45. Kruchten, P.: The Rational Unified Process: An Introduction. (3rd edition). Addison-Wesley, USA, (2003)
46. Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK– 4th edition). Project Management Institute, USA, (2008)
47. Návrat, P. et al.: Artificial Intelligence, 2002, Slovak University of Technology in Bratislava. STU Press, (2002)
48. Kvasnička, V., Pospíchal, J.: Mathematical Logic. Slovak University of Technology in Bratislava. STU Press, (2005)
49. Baader, F., Horrocks, I., Saatler, U.: Description Logics. In Steffen Staab and Rudi Studer, editors, Handbook on Ontologies, International Handbooks on Information Systems, Springer. 3-28, (2004)
50. Líška, M., Návrat, P.: OWL4SPEM – A semantic framework for software process engineering, (2010). [Online]. Available: http://www.ontologia.sk/owl4spem/ (accessed February 22, 2011)
51. Habala O., Paralič M., Rozinajová V., and Bartalos P.: Semantically-Aided Data-Aware Service Workflow Composition. In: M. Nielsen et al. (Eds.): SOFSEM 2009, LNCS 5404, pp. 317–328, 2009, Springer-Verlag Berlin Heidelberg 2009
52. Bartalos, Peter - Bieliková, Mária: QoS Aware Semantic Web Service Composition Approach Considering Pre/Postconditions. In: IEEE ICWS 2010, Eighth International Conference on Web Services, Miami, Florida, 5-10 July 2010 : Proceedings. - Los Alamitos : IEEE Computer Society, 2010. - ISBN 978-0-7695-4128-0. - pp. 345-35254.

53. Bartalos, P. Effective Automatic Dynamic Semantic Web Service Composition. Information Sciences and Technologies, Bulletin of the ACM Slovakia, Vol. 3, No. 1, (2011)

Miroslav Líška received the M.S. degree in informatics from the Technical University in Košice, Slovakia in 2002, and the PhD. degree in software and information systems from the Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava in 2010. His interests include semantic web, ontologies, software process engineering and semantic enterprise oriented architectures. He currently works as a semantic web architect in Datalan, in Bratislava, Slovakia.

Pavol Navrat received his Ing. (Master) cum laude in 1975, and his PhD. degree in computing machinery in 1984 both from Slovak University of Technology in Bratislava. He is currently a professor of Informatics at the Slovak University of Technology and serves as the director of the Institute of Informatics and Software Engineering. During his career, he was also with other universities abroad. His research interests include related areas from software engineering, artificial intelligence, and information systems. He published numerous research articles, several books and co-edited and co-authored several monographs. Prof. Navrat is a Fellow of the IET and a Senior Member of the IEEE and its Computer Society. He is a Senior Member of the ACM and chair of the ACM Slovakia Chapter. He is also a member of the Association for Advancement of Artificial Intelligence, Slovak Society for Computer Science and Slovak Artificial Intelligence Society. He serves on the Technical Committee 12 Artificial Intelligence of IFIP as the representative of Slovakia.

Received: December 30, 2010; Accepted: March 02, 2011.