Ontology to Classify Learning Material in Software Engineering Knowledge Domain

Joselaine Valaski, Andreia Malucelli, Sheila Reinehr, Ricardo Santos

Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba – PR – Brasil

{jvalaski, malu}@ppgia.pucpr.br, sheila.reinehr@pucpr.br, ricardo.c.r.santos@gmail.com

***Abstract**. This paper proposes an ontology to automatic classification of learning materials to the Software Engineering knowledge domain. The Software Engineering Body of Knowledge (SWEBOK) was used to define the hierarchical structure of the knowledge area. The Rational Unified Process (RUP) was used to add the axioms to represent the relationships between concepts and to enable the reasoning to SWEBOK knowledge areas. Two testing scenarios were designed and experiments were performed. The results show that the ontology is able to classify and locate learning materials from the Software Engineering area, according to the desired area, role, artifact or task.*

1. Introduction

The development of new web-based technologies has increased the number of learning environments, from simple learning resources repositories to more complex learning environments. In these environments, learner can access information, communicate among themselves and learn in a self-learning method [Ruiz et al. 2008].

This self-learning process can happen through many didactic materials, such as digital books, slideshows, audio or video recordings, etc. These materials allow knowledge sharing within a common interest domain and are available to anyone, anytime, anywhere. This can facilitate the learning of subjects that require highly trained professionals, who need to be up-to-date with the state of the art of technology. Software Engineering can be named as one of such subjects.

However, the self-learning environment can present challenges that hinder the real knowledge acquisition. The difficulty to search the learning materials according to the learning theme is one of these challenges. This search can be more difficult to learners due to the range of knowledge themes [Yu 2010], making the identification of desired learning materials a challenge [Fischer 2001].

The process of classifying learning materials according to their knowledge area can be an alternative to facilitate their retrieval. However, these classification mechanisms must use a common language that would allow knowledge sharing to occur effectively [Davenport and Prusak 1998].

Most knowledge areas have terminology problems in the use of consensual terms, as an example, the Software Engineering area. It is common that different development teams use diverse terms for the same concepts. Even though many software engineers work with Software Engineering, some professionals claim to never have studied the subject [Wongthongtham 2006]. Thus, it is likely that professionals find some difficulty to search adequate learning materials due to lack of a common terminology.

In this context, ontologies play an important role because they can be applied to provide a common shared understanding of an information structure among individuals or organizations, as well as be used to enable the knowledge domain reuse and make explicit assumptions of a domain [Noy and McGuinness 2010].

Ontologies can describe a hierarchy of concepts related by subsumption relationships, in this case, a taxonomy-driven concept; or a structure, where the axioms are added in order to express relationships between concepts and to restrict their intentional interpretations [Guarino 1998]. Through ontologies, hierarchical structures of themes related to the learning materials can be defined using a common vocabulary to the knowledge area. Furthermore, it is possible to add reasoning to this structure in order to help the automatic classification of learning materials within the defined hierarchy. The automated classification is relevant when people do not hold enough knowledge to identify the theme related to the learning materials due to lack of common vocabulary of the knowledge area. Software engineers can be mentioned as an example.

In this context, this paper aims to propose an ontology to automatic classification of learning materials related to the Software Engineering knowledge area. The ontology aims to facilitate the search for learning materials within the given domain. The Software Engineering Body of Knowledge (SWEBOK) [Abran and Moore 2004] was used to define the hierarchical structures of knowledge. The SWEBOK is intended to reach broad consensus on the area of Software Engineering [Sicilia 2005]. The Rational Unified Process (RUP) was used to add axioms to represent the relationships between concepts and enable the reasoning to the SWEBOK knowledge area.

The remainder sections of this paper are organized as follows: Section 2 presents the related work; Section 3 describes in details the proposed ontology; in Section 4 some experiments are discussed; Section 5 concludes the paper.

2. Related Works

There are several papers proposing ontologies for the Software Engineering area. This section presents these researches and their approaches.

Mendes and Abran (2005) present a prototype of an ontology to represent the domain of Software Engineering, based on the SWEBOK guide. A literal extraction from the guide results in approximately 4,000 concepts. In this approach, there is no intention to establish a hierarchical structure of the Software Engineering knowledge area. Sicilia et al. [2005] also proposes a SWEBOK based ontology with a descriptive part in order to identify artifacts and activities and a prescriptive part, with approaches and concrete activities'rules for "commonly accepted" practical activities. Hilera et al. (2005) propose an ontology called OntoGLOSE based on the Software Engineering Terminology Glossary, published by IEEE. OntoGLOSE includes about 1,500 concepts, corresponding to 1,300 glossary terms with their different meanings.

More specific approaches are established on the Software Engineering domain as well. The Win-Win approach represents a model created to manage the necessary collaboration and negotiation by the people involved in the software lifecycle stage [Bose 1995]. ONTODM represents the knowledge of requisite specification techniques of a multi-agent systems family in an application domain. It is being used as a CASE tool to help to elicit and specify the domain models. [Girardi and Faria 2003]. Sánchez et al. [2005] propose an ontology to represent the different meanings of the term model, incorporating the different concepts related to the terms. Cyc [2011] presents a UML subOntology integrated in the OpenCyc ontology containing about 100 concepts, 50 relationships and 30 instances, including UMLModel Element, UMLClassifier, UMLClass and UMLStateMachine, according to SWEBOK's Software Projects Notations subarea, from the Software Project area. The XCM ontology provides a pattern to a component definition that appears in different component models and standardizes these differences [Tansalarak and Claypool, 2004]. Deridder [2002] presents a general ontology on concepts related to software maintenance. An ontology organized in five subontologies to represent the knowledge related with software systems, the necessary skills to software maintainers, with maintenance process activities, organizational maintenance topics and tasks that constitute any application domain is proposed by Dias et al. [2003]. Ruiz et al. [2004] propose an ontology composed by four subontologies: products, activities, organization processes and agents. Vizcaino et al. [2005] propose an ontology composed by the ontologies proposed by Deridder [2002], Dias et al. [2003] and Ruiz et al. [2004]. The propose of Deridder [2002], Dias et al. [2003], Ruiz et al. [2004] and Vizcaino et al. [2005] are based on an initial software maintenance ontology proposed by Kitchenham et al. [1999]. Boehm and In [1996] propose an ontology with concepts related to software quality attributes and information about the software architectures influences and development processes on these attributes Other ontology related to software process concepts is proposed by Falbo et al. [2002]. An ontology with the software measurement terminology, associated with fundamental concepts is proposed by Garcia et al. [2005]. Tautz and Greese [1998] present an ontology of the GQM (Goal Question Metric) paradigm, and an ontology with concepts related to software process, including Life Cycle Models concepts, Software Processes, Activities, Procedures, Tasks, Roles or Artifacts is presented by Falbo et al. [1998]. The SPOnt, an ontology that reused concepts from other ontologies related to decision support systems, establishing relationships, is proposed by Larburu et al. [2003]. González-Pérez and Henderson-Sellers [2006] present an ontology for software development methodology that include a metamodel and an architecture divided into three domains. Lin et al. [2003] propose an ontology for the IEEE 12207 and the CMMI Standards that can be applied in an organization in order to inspect and enhance the software processes maturity. An ontology particularly focused on the Software Engineering area was developed by Wongthongtham et al. (2007), the first Software Engineering oriented ontology, based on the SWEBOK's areas of knowledge. This ontology presents only a hierarchical structure; it does not use axioms to define the concepts related to the knowledge areas.

There are several proposals for ontologies in the Software Engineering area, however, there is not an ontology to classify materials according to the Software Engineering knowledge area. The next section discusses the proposal of an ontology to help solving this problem.

3. Proposed Ontology

This section presents an ontology composed by SWEBOK and RUP concepts to classify learning materials in the Software Engineering knowledge area. The ontology was developed with the ontology editor Protégé [Stanford 2011].

To define the knowledge's hierarchical structure related to Software Engineering, the SWEBOK´s definition knowledge area was used. The SWEBOK is a guide created under the patronage of the Institute of Electrical and Electronics Engineers (IEEE) with the objective of serving as reference to Software Engineering related subjects [Abran and Moore 2004]. This guide presents a hierarchical classification of the Software Engineering topics, where the higher level is the knowledge areas.

However, the definition of a hierarchical structure is not enough to allow the automatic classification of learning materials according to the defined structure. The SWEBOK does not present an approach to the definition of their knowledge areas using relationships among the concepts or explicit properties. For this reason, RUP was also used. RUP presents well-defined relationships among the main concepts, which are: Discipline, Artifact, Role and Task. Although RUP is a software development process, hence, not exactly focused on knowledge areas, the concept of disciplines can be related between some SWEBOK knowledge areas, as shown in Table 1. In this proposal, only the areas with total correspondence were mapped.

In the following subsections the details of the proposed ontology for RUP and the integration of this ontology with the ontology for the classification of learning materials according to the SWEBOK knowledge areas are presented.

Table 1 – Relationship between the SWEBOK areas and RUP disciplines

SWEBOK Area	RUP Discipline
Software Engineering Management	Project Management
Software Engineering Process	
Software Engineering Tools and Methods	
Software Configuration Management	Configuration and Change Management
Software Construction	Implementation
Software Design	Analisys and Design
Software Maintenance	
Software Quality	
Software Requirements	Business Modeling Requirements
Software Testing	Test
	Deployment
	Environment

3.1 OntoRUP: RUP representation ontology

OntoRUP was developed according to the Artifact, Role and Task concepts and their relationships with the Discipline concept. Through these four concepts and their relationships, classes and their properties were created. Table 2 presents the created classes and properties.

Table 2 – Classes and properties from OntoRUP

Domain Class	Range Class	Property	Special Property (inverse)
Artifact Task	Discipline	hasDomain	isDomainOf
Discipline	Artifact Task	isDomainOf	hasDomain
Role	Artifact	modify	isModified
Artifact	Role	isModified	modify
Task	Role	hasPerformer	isPerformerOf
Role	Task	isPerformerOf	hasPerformer

The general proposed hierarchy is presented in Figure 1. The RupElements class was created in order to group the derivative concept classes: Discipline, Artifact, Role and Task concepts.

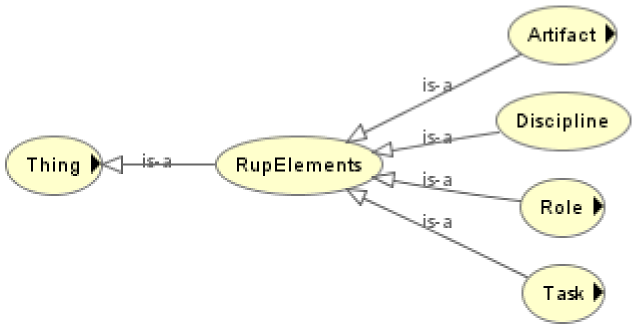
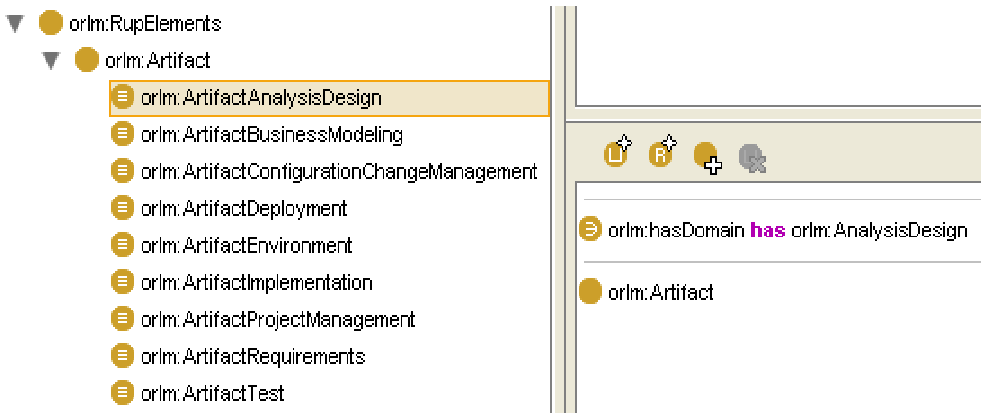


Figure 1 – OntoRUP general hierarchy

The Discipline class was created to represent the nine disciplines that compose the RUP model. Through this class the other relationships are established and then the integration is done with the SWEBOK´s knowledge areas.

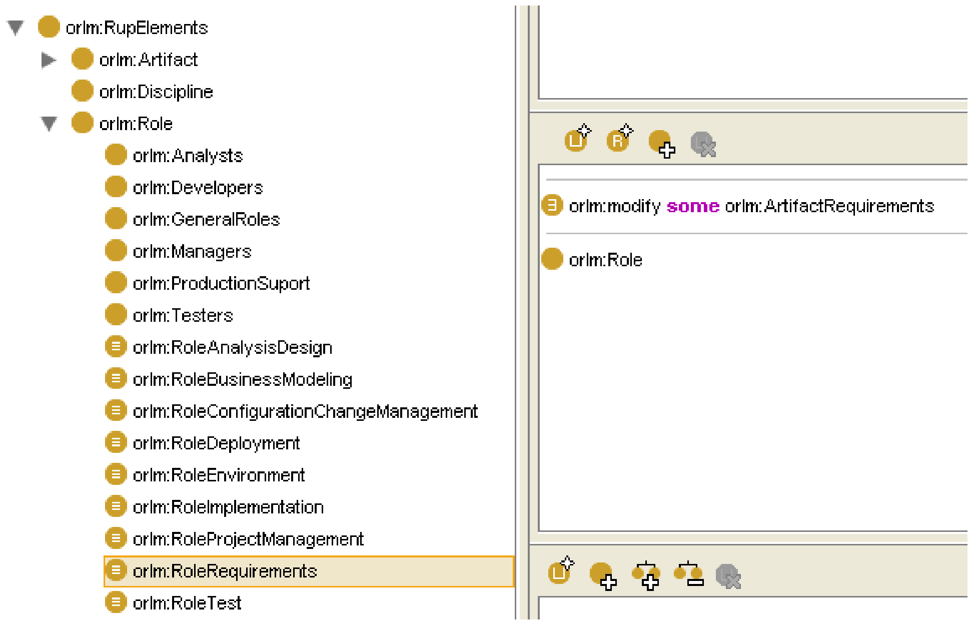
The Artifact class was created to represent the software artifacts that are used within the RUP process. The Artifact class is directly related to the Discipline class through the hasDomain property. According to this relationship, subclasses were created, that identify the artifacts related to each of the nine disciplines proposed in the RUP model. Figure 2 presents an example of the hasDomain property.


orlm:RupElements
orlm:Artifact
orlm:ArtifactAnalysisDesign
orlm:ArtifactBusinessModeling
orlm:ArtifactConfigurationChangeManagement
orlm:ArtifactDeployment
orlm:ArtifactEnvironment
orlm:ArtifactImplementation
orlm:ArtifactProjectManagement
orlm:ArtifactRequirements
orlm:ArtifactTest

orlm:hasDomain **has** orlm:AnalysisDesign
orlm:Artifact

Figure 2 – hasDomain property

The Role class was created to represent the corresponding subclasses to the six groups of roles within the RUP, namely: Analysts, Developers, General Roles, Manager, Production Support and Testers. Furthermore, within the Role class, corresponding subclasses of the roles related to each of the nine disciplines were also created as shown in Figure 3. To establish the relationship between the Role and Discipline classes, it was used the property "modify" that relates the Role class to the Artifact's subclasses. As the subclasses of Artifact are already related to the Discipline class, the relationship between the Role and Discipline classes is also completed.


orlm:RupElements
orlm:Artifact
orlm:Discipline
orlm:Role
orlm:Analysts
orlm:Developers
orlm:GeneralRoles
orlm:Managers
orlm:ProductionSuport
orlm:Testers
orlm:RoleAnalysisDesign
orlm:RoleBusinessModeling
orlm:RoleConfigurationChangeManagement
orlm:RoleDeployment
orlm:RoleEnvironment
orlm:RoleImplementation
orlm:RoleProjectManagement
orlm:RoleRequirements
orlm:RoleTest

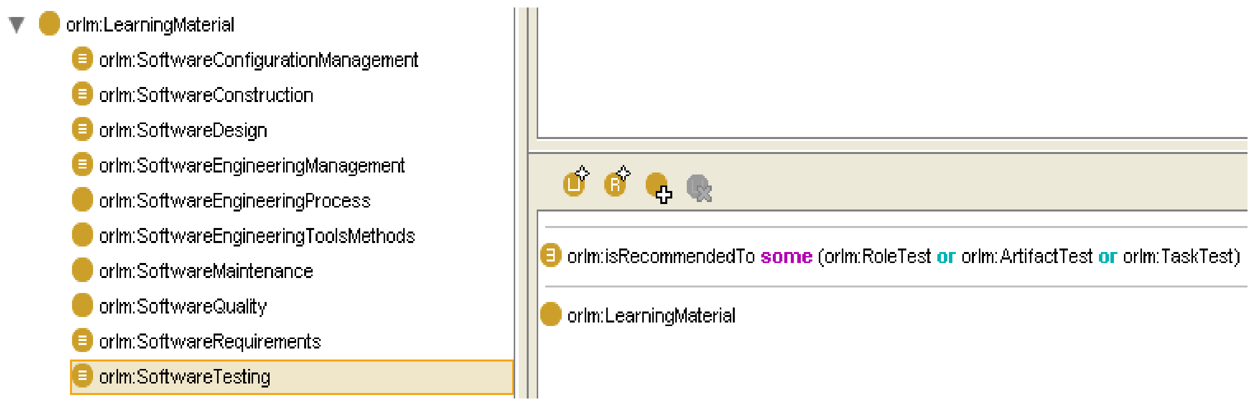
orlm:modify **some** orlm:ArtifactRequirements
orlm:Role

Figure 3 – Role's subclasses

The Task class was created to represent the tasks of the RUP model. The Task class has direct relationship with the Discipline class through the hasDomain property. Based on this relationship have been created subclasses to represent the tasks corresponding to each of the nine RUP disciplines specified in the model.

3.2 Software Engineering Learning Materials Ontology

Once established the ontology structure for representation of RUP elements, it was defined the necessary elements to enable the classification of learning materials within the Software Engineering domain. The LearningMaterial class was created to represent the learning materials, and its subclasses were created based on the ten SWEBOK's areas, as shown in Figure 4.


orlm:LearningMaterial
orlm:SoftwareConfigurationManagement
orlm:SoftwareConstruction
orlm:SoftwareDesign
orlm:SoftwareEngineeringManagement
orlm:SoftwareEngineeringProcess
orlm:SoftwareEngineeringToolsMethods
orlm:SoftwareMaintenance
orlm:SoftwareQuality
orlm:SoftwareRequirements
orlm:SoftwareTesting

orlm:isRecommendedTo some (orlm:RoleTest or orlm:ArtifactTest or orlm:TaskTest)
orlm:LearningMaterial

Figure 4 – Learning Materials according to the SWEBOK

In order to define the ten areas of the SWEBOK using explicit and formal properties, the defined concepts of RUP ontology was used. It is possible to identify the related discipline through any of the Artifact, Role and Task concepts, and through mapping it is possible to know the SWEBOK´s knowledge area. Because of that, the isRecommendedTo property was created, as shown in Table 3, in order to be able to recommend a learning material related to any of the three concepts presented in RUP. Thus, when adding a learning material it is possible: to recommend the material for the use of a specific artifact, such as a Business Case; the execution of a specific task, such as Architectural Analysis; or the execution of a specific role, such as System Analyst.

Table 3 – isRecommendedTo property

Domain Class	Range Class	Property	Special Property (inverse)
LearningMaterial	Artifact Task Role	isRecommendedTo	hasRecommendation
Artifact Task Role	LearningMaterial	hasRecommendation	isRecommendedTo

Through the related recommendation it is possible to classify the material according to the SWEBOK's knowledge areas. For instance, a learning material will be classified as belonging to the Test knowledge area, if it has the isRecommendedTo property related to, at least, one instance of the Artifact, Role or Task classes, linked to the Test discipline.

These possibilities of recommendations can help to obtain a more accurate classification of the learning material, especially when there is no formal knowledge regarding to which knowledge area the material belongs to.

4. Results

The ontology was proposed to be applied in a self-learning environment where people share their knowledge related to the Software Engineering area by adding learning materials. The proposed ontology will help in the classification of learning materials, mainly because software engineers may not use a common vocabulary or may not have enough knowledge to classify correctly the material within the appropriate domain. Furthermore, the ontology will facilitate the recommendation of these learning materials.

Two scenarios were designed to verify the proposal's viability. The scenario 1 was used to test the classification of learning materials and scenario 2 was used to test the recommendation of these materials. The simulations were created using the Protégé tool.

- Scenario 1 – Learning Materials Classification

Instances of learning materials were added using the Protégé tool, as shown in Figure 5. Also recommendations were made through the isRecommendedTo property. Each recommendation was associated with instances of Artifact, Role or Task classes.

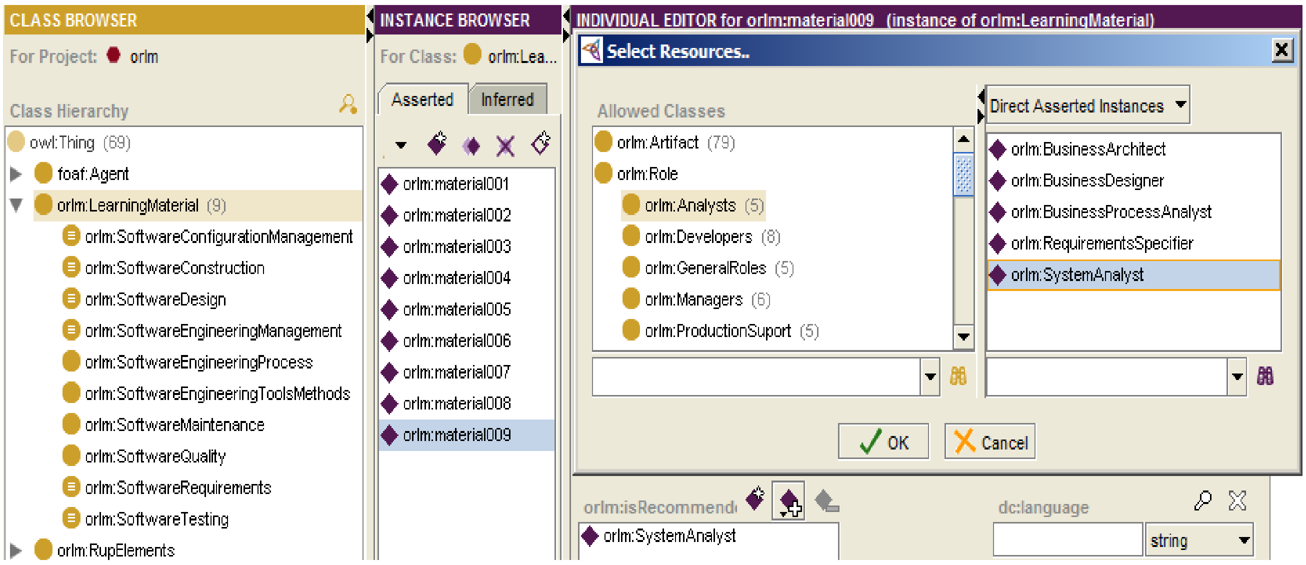


Figure 5 – Learning materials instances included using Protégé

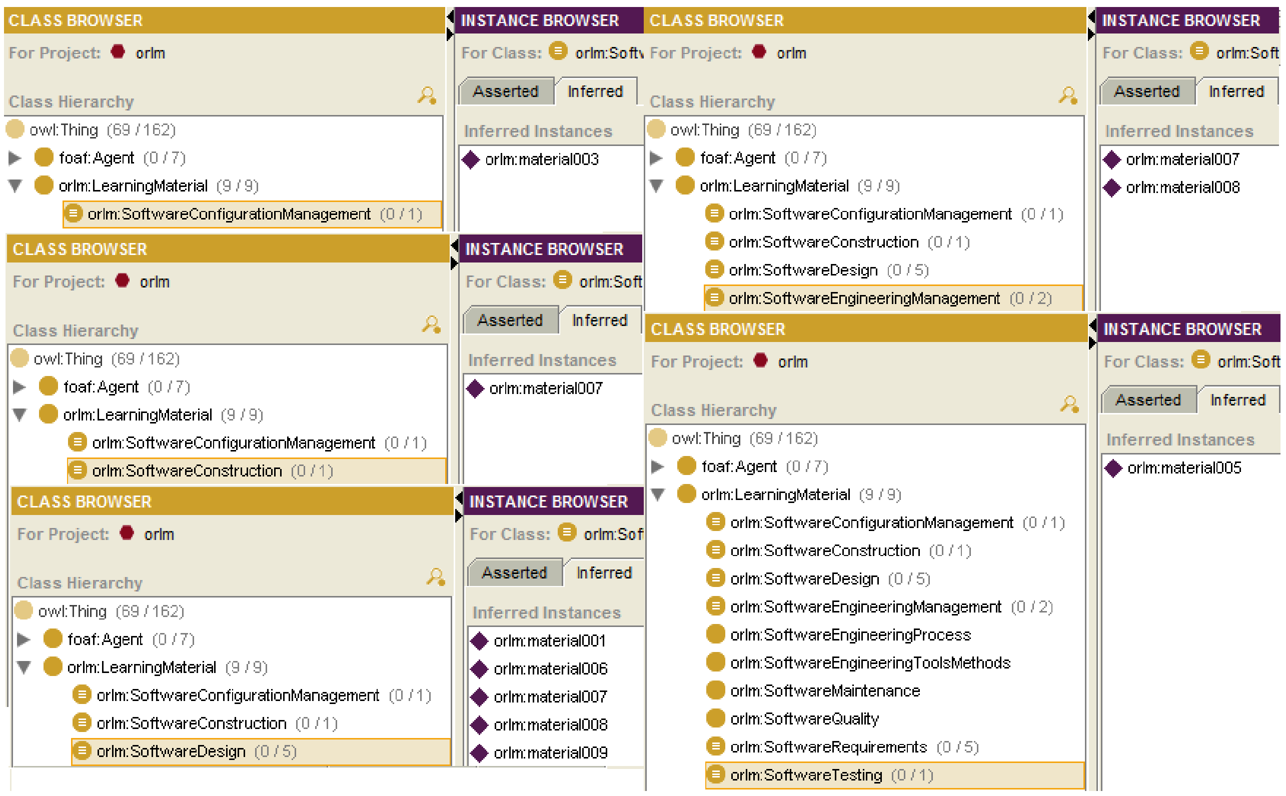
The values assigned to the isRecommendedTo class for each one of the learning materials are shown in Table 4.

Table 4 – Values assigned to the isRecommendedTo property

Id. Material	Recommendation for Artifact	Recommendation for Role	Recommendation for Task
material001	Analisys Model Use Case Model	System Analyst	
material002		Requirements Specifier	
material003			Create Baseline
material004		System Administrator	
material005	Test Plan		
material006			Architectural Analisys
material007		Software Architect	
material008	Business Case	System Analyst	
material009		System Analyst	

The Pellet reasoned, version 1.5.2, was used to classify the learning materials. As shown in Figure 6, it is possible to verify that the ontology correctly classified the learning materials according to the defined concepts.

However, it is important to provide mechanisms to help software engineer to make their recommendations in order to avoid inconsistencies. For instance, the material identified as "material008", was recommended to be used in the Business Case artifact. In this case, it should not be possible to recommend it for the System Analyst role, as this role has no relationship with this artifact. As a result, the material was classified in three knowledge areas, one of them due to artifact recommendation, and the other two due to recommendation by role. The ontology proposed can be used to help filter consistent recommendations among Artifact, Role and Task classes.



CLASS BROWSER
For Project: orlm
Class Hierarchy
owl:Thing (69 / 162)
foaf:Agent (0 / 7)
orlm:LearningMaterial (9 / 9)
orlm:SoftwareConfigurationManagement (0 / 1)

INSTANCE BROWSER
For Class: orlm:Softv
Asserted | Inferred
Inferred Instances
orlm:material003

CLASS BROWSER
For Project: orlm
Class Hierarchy
owl:Thing (69 / 162)
foaf:Agent (0 / 7)
orlm:LearningMaterial (9 / 9)
orlm:SoftwareConfigurationManagement (0 / 1)
orlm:SoftwareConstruction (0 / 1)

INSTANCE BROWSER
For Class: orlm:Soft
Asserted | Inferred
Inferred Instances
orlm:material007

CLASS BROWSER
For Project: orlm
Class Hierarchy
owl:Thing (69 / 162)
foaf:Agent (0 / 7)
orlm:LearningMaterial (9 / 9)
orlm:SoftwareConfigurationManagement (0 / 1)
orlm:SoftwareConstruction (0 / 1)
orlm:SoftwareDesign (0 / 5)

INSTANCE BROWSER
For Class: orlm:Sof
Asserted | Inferred
Inferred Instances
orlm:material001
orlm:material006
orlm:material007
orlm:material008
orlm:material009

CLASS BROWSER
For Project: orlm
Class Hierarchy
owl:Thing (69 / 162)
foaf:Agent (0 / 7)
orlm:LearningMaterial (9 / 9)
orlm:SoftwareConfigurationManagement (0 / 1)
orlm:SoftwareConstruction (0 / 1)
orlm:SoftwareDesign (0 / 5)
orlm:SoftwareEngineeringManagement (0 / 2)

INSTANCE BROWSER
For Class: orlm:Soft
Asserted | Inferred
Inferred Instances
orlm:material007
orlm:material008

CLASS BROWSER
For Project: orlm
Class Hierarchy
owl:Thing (69 / 162)
foaf:Agent (0 / 7)
orlm:LearningMaterial (9 / 9)
orlm:SoftwareConfigurationManagement (0 / 1)
orlm:SoftwareConstruction (0 / 1)
orlm:SoftwareDesign (0 / 5)
orlm:SoftwareEngineeringManagement (0 / 2)
orlm:SoftwareEngineeringProcess
orlm:SoftwareEngineeringToolsMethods
orlm:SoftwareMaintenance
orlm:SoftwareQuality
orlm:SoftwareRequirements (0 / 5)
orlm:SoftwareTesting (0 / 1)

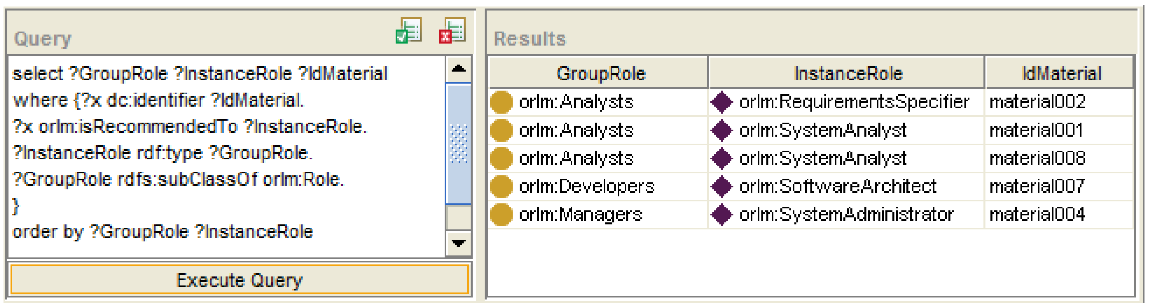
INSTANCE BROWSER
For Class: orlm:Soft
Asserted | Inferred
Inferred Instances
orlm:material005

Figure 6 – Learning Materials classification using Pellet

- Scenario 2 – Learning Materials Recommendation

Scenario 2 was designed to present the possible recommendations of the learning materials once these materials will be available in a learning environment. According to the simulation described in scenario 1, after the learning materials were classified using the inference mechanisms, it is possible to search for these materials through the knowledge areas defined in SWEBOK. For instance, it is possible to retrieve all the learning materials related to the Software Requirement area. However, besides retrieving the materials by Software Engineering knowledge area, the ontology also allows to find all the materials according to recommendations, by Artifact, Role or Task.

SPARQL was used to simulate a preview of these possibilities. The SPARQL is a language to retrieve data from Web Ontology Language (OWL) files. Figure 7 presents a SPARQL query in order to retrieve learning materials recommended by Roles. In this case, the learning materials are retrieved through the Roles view; however, the queries can be executed by Artifacts and Tasks as well.



Query

```
select ?GroupRole ?InstanceRole ?IdMaterial
where {?x dc:identifier ?IdMaterial.
?x orlm:isRecommendedTo ?InstanceRole.
?InstanceRole rdf:type ?GroupRole.
?GroupRole rdfs:subClassOf orlm:Role.
}
order by ?GroupRole ?InstanceRole
```

Execute Query

Results

GroupRole	InstanceRole	IdMaterial
orlm:Analysts	orlm:RequirementsSpecifier	material002
orlm:Analysts	orlm:SystemAnalyst	material001
orlm:Analysts	orlm:SystemAnalyst	material008
orlm:Developers	orlm:SoftwareArchitect	material007
orlm:Managers	orlm:SystemAdministrator	material004

Figure 7 – Query using SPARQL

It is important to point out that new recommendations may be added to the learning materials according to their use. For example, a learning material that was added with the System Analyst role may also be recommended to the Elicit Stakeholder Requests task. So, the level of details for the recommendation is enhanced and the retrieval of material becomes more precise.

5. Conclusion

This paper presented an ontology to automatically classify learning materials related to the Software Engineering knowledge area, aiming to facilitate the search for these materials.

The ontology was defined using the main structure of ten SWEBOK knowledge areas and the concepts and relationships among Artifact, Task and Role elements from RUP model. RUP was used to define SWEBOK knowledge areas through axioms to enable the automatic classification of learning materials according to recommendations.

Some experiments were performed and it was possible to conclude that the ontology classifications were correctly, according to the Software Engineering knowledge areas. Furthermore, the ontology provides views of the learning materials under three aspects, recommendations by artifacts, tasks and role. This diversity can be another facilitator for retrieving the desired material.

The proposed ontology will be integrated to a self-learning environment, and experiments with Software Engineering students and professionals will be performed in order to evaluate the proposal.

References

Abran, A. and Moore, J. W. (2004). “SWEBOK - Guide to the Software Engineering Body of Knowledge”. IEEE CS Professional Practices Committee.

Boehm, B. and In, H. (1996). “Identifying Quality Requirements Conflicts”. IEEE Software, pp. 25–35.

Bose, P. (1995). “Conceptual design model based requirements analysis in the Win-Win framework for concurrrent requirements engineering”. In: IEEE Workshop on Software Specification and Design (IWSSD).

Clemente, J., Ramírez, J. and Antonio, A. (2010). “A proposal for student modeling based on ontologies and diagnosis rules”. Expert Systems with Applications, pp. 8066-8078.

Cyc (2011). Cyc: OpenCyc.org: Formalized Common Knowledge. Cycorp, USA. http://www.opencyc.org, April.

Davenport, T.H. and Prusak, L. (1998). “Working Knowledge: How Organizations Manage What They Know”. Harvard Business School Press.

Deridder, D. (2002). “A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities”. In: 5th Joint Conference on Knowledge-Based Software Engineering (JCKBSE), Maribor, Slovenia.

Dias, M.G., Anquetil, N., and Oliveira, K.M. (2003). “Organizing the Knowledge Used in Software Maintenance”. In: Journal of Universal Computer Science, pp. 641–658.

Falbo, R., Menezes, C. and Rocha, A. (1998). “Using Ontologies to Improve Knowledge Integration in Software Engineering Environments”. In: 4th International Conference on Information Systems Analysis and Synthesis(ISAS), Orlando, USA.

Falbo, R.A., Guizzardi, G., Duarte, K.C. (2002).”An Ontological Approach to Domain Engineering”. In: Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, pp. 351–358.

Fischer, G. (2001). “User Modeling in Human–Computer Interaction”. In: User modeling and user-adapted interaction, pp. 65–86.

García, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M. and Genero, M. (2006). “Towards a consistent terminology for software measurement”. Information and Software Technology. pp. 631-644.

Girardi, R. and Faria, C. (2003). “A Generic Ontology for the Specification of Domain Models”. In: Proceedings of 1st International Workshop on Component Engineering Methodology (WCEM’03) at Second International Conference on Generative Programming and Component Engineering, Erfurt, Germany.

González-Pérez, C. and Henderson-Sellers, B. (2006). “An Ontology for Software Development Methodologies and Endeavours”. Ontologies for Software Engineering and Technology, Springer-Verlag, Berlin.

Hilera, J.R., Sánchez-Alonso, S., García, E. and Del Molino, C.J. (2005). “OntoGLOSE: A Light-weight Software Engineering Ontology”. In: 1st Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain.

Kitchenham, B.A., Travassos, G.H., Mayrhauser, A., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R. and Yang, H. (1999). “Towards an Ontology of Software Maintenance”. Journal of Software Maintenance: Research and Practice, pp. 365–389.

Larburu, I.U., Pikatza, J.M., Sobrado, F.J., García, J.J. and López, D. (2003). “Hacia la implementación de una herramienta de soporte al proceso de desarrollo de software”. In: Workshop in Artifficial Intelligence Applications to Engineering (AIAI), San Sebastián, Spain.

Lin, S., Liu, F. and Loe, S. (2003). “Building A Knowledge Base of IEEE/EAI 12207 and CMMI with Ontology”. In: Sixth International Protégé Workshop, Manchester, England.

Mendes, O. and Abran, A. (2005). “Issues in the development of an ontology for an emerging engineering discipline”. In: First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain.

Noy, N. F. and McGuinness, D. L. (2001). “Ontology Development 101: A Guide to Creating Your First Ontology”. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

Ruiz, F., Vizcaíno, A., Piattini, M. and García, F. (2004). “An Ontology for the Management of Software Maintenance Projects”. International Journal of Software Engineering and Knowledge Engineering, pp. 323–349.

Ruiz, M., Diaz, M., Soler, F. and Perez, J. (2008). “Adaptation in current e-learning systems”. Computer Standards & Interfaces, pp. 62-70.

Sánchez, D.M., Cavero, J.M. and Marcos, E. (2005). “An ontology about ontologies and models: a conceptual discussion.” In: First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain.

Sicilia, M., Cuadrado, J. J., Garcia, E., Rodriguez, D. and Hilera, J. R. (2005). “The evaluation of ontological representation of the SWEBOK as a revision tool”. In: 29th Annual International Computer Software and Application Conference (COMPSAC), Edinburgh, UK, pp. 26–28.

Stanford (2011). “The Protégé Ontology Editor and Knowledge Acquisition System”, http://protege.stanford.edu/index.html, April.

Tansalarak, N., Claypool and K.T. (2004). “XCM: A Component Ontology.” In: Workshop on Ontologies as Software Engineering Artifacts (OOPSLA), Vancouver, Canada.

Tautz, C. and Von Wangenheim, C.(1998). “REFSENO: A Representation Formalism for Software Engineering Ontologies”. Fraunhofer IESEReport No. 015.98/E, version 1.1, October 20.

Vizcaíno, A., Anquetil, N., Oliveira, K., Ruiz, F. and Piattini, M. (2005). “Merging Software Maintenance Ontologies: Our Experience”. In: First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcala de Henares, Spain.

Yu, Z., Zhou, X. and Shu, L. (2010). “Towards a semantic infrastructure for context-aware e-learning”. Multimedia Tools and Applications, pp. 71–86.

Wongthongtham, P. (2006). “A methodology for multi-site distributed software development.” PhD Thesis, Curtin University of Technology.