Applications of Software Engineering Ontology

Pornpit Wongthongtham
Digital Ecosystems and Business Intelligence Institute,
Curtin University of Technology,
Perth, WA 6102, AUSTRALIA
E-mail: P.Wongthongtham@cbs.curtin.edu.au,

Natsuda Kasisopha
Digital Ecosystems and Business Intelligence Institute,
Curtin University of Technology,
Perth, WA 6102, AUSTRALIA
E-mail: natsuda.kasisopha@postgrad.curtin.edu.au,

Surasak Komchaliaw
Digital Ecosystems Business and Intelligence Institute,
Curtin University of Technology,
Perth, WA 6102, AUSTRALIA
E-mail: maxtor_p@hotmail.com.

Abstract

Software engineering ontology (SE Ontology) has been developed initially to support multi-site distributed software development. It tackles the disadvantages associated with remote communication and co-ordination for software engineering projects in internationally multi-site software development environments. The SE Ontology defines common shareable software engineering knowledge and typically provides software engineering concepts: what the concepts are, how they are related, and why they are related. When this generic ontology is specialized to a particular project and populated with instances which reflect the project information it provides this common understanding of project information to all the distributed members of a development team in a multi-site development environment. In this paper we present some examples of the SE Ontology applications through multi-site software development processes.

Key Words: Software engineering ontology, Ontology evolution, Multi-site software development

1. Introduction

Software engineering ontology (SE Ontology) has been developed initially to support multi-site distributed software development. It tackles the disadvantages associated with remote communication and co-ordination for software engineering projects in internationally multi-site software development environments. Different teams might not be aware of what tasks are being carried out by others, potentially leading to problems such as two groups overlapping in some work or work not being performed due to a misinterpretation of the task. Wrong tasks may be carried out due to ignorance of who to contact to get the proper details. If everyone working on a certain project is located in the same area, then situational awareness is relatively straightforward; however for distributed international development, the overheads in communications to get together to discuss the problems, to raise issues, to make decisions and to find answers are very high. Consequently, these problems cause developmental delays and software problems as outstanding issues are not resolved and issues cannot be discussed immediately or in time, over a distributed team environment.

The SE Ontology defines common shareable software engineering knowledge and typically provides software engineering concepts: what the concepts are, how they are related, and why they are related [1]. When this generic ontology is specialized to a particular project and populated with instances which reflect the project information it provides this common understanding of project information to all the distributed members of a development team in a multi-site development environment [2].

In this paper we present some examples of the SE Ontology applications through multi-site software development processes. We firstly provide a concise description of the SE Ontology and depict its evolution in section 2. Secondly we give some examples of the applications in section 3. We finally conclude the paper in section 4.

2. Software engineering ontology and its evolution

The SE Ontology which is available online at www.seontology.org represents the knowledge by structuring concepts, their relationships and their constraints, i.e. it is an abstract level of representation. Project data over a period of time will be populated as instances in the deployment stage. Precisely stated, the SE Ontology captures the generic software engineering concepts as well as the specific software engineering project information / data / agreements. The key ingredients that make up the SE Ontology are a vocabulary of basic software engineering terms and a precise specification of what those terms mean. The SE Ontology is populated with specific instances for a particular project for the corresponding software engineering concepts. These instances contain the actual project data being queried in the knowledge based applications. Thus the SE Ontology includes the set of actual project data (i.e. instances of the concepts) and assertions that the instances are related to each other according to the specific relations between the concepts.

We consider evolution of the SE Ontology in two senses i.e. (i) evolution in generic software engineering concepts (concepts level) which reflects new kinds of software, application, or system being developed on broader and different understandings (ii) evolution in specific project data (instances level) which reflects project development, changes in the software requirements or in the design process in order to incorporate additional functionality to software, application, or system or to allow incremental improvement and the like.

In this paper we use Description Logic (DL) to logically formalize the SE Ontology which is written in Ontology Web Language (OWL). OWL exploits many of the strengths of DL including well defined semantics and practical reasoning techniques [3].

3. Applications of software engineering ontology

In this section, we present concrete approach of using the SE Ontology. In the presentation, we follow software development processes i.e. requirements, design, construction and testing. In each process we describe concerned issues in multi-site software development followed by a description of the SE Ontology and solutions of using the SE Ontology respectively.

3.1 Requirements

Issues in multi-site software development

In multi-site software development situations, analyst team gathers desired system functionality from clients and share understanding of the problem domain and system specification to design and implementation teams who may be physically far away. The physical distance could become a crucial issue due to different understanding led to ambiguous, continually evolving, incomplete specifications. There may be a major rework after system implementation. Thus it is essential to assure that all remote team members have shared understanding of the problem definition. In case of client's objectives change it is important to assure that all remote team members are aware of change of requirements.

Description of the SE Ontology

The SE Ontology formally represents requirement knowledge. It captures type of requirements e.g. functional requirements, non-functional requirements, system requirements, or user requirements, and so on. It also confines requirement analysis and elicitation. Additionally it describes requirements specification document. Below are some examples of requirement knowledge captured in the SE Ontology:

- Functional_Requirement $\subseteq$ Requirement
- Architectural_Model $\subseteq$ Conceptual_Model $\subseteq$ Requirement_Analysis
- Prototypes $\subseteq$ Elicitation_Techniques $\subseteq$ Requirement_Elicitation
- ObjectProperty(hasRequirement domain(Requirement_Specification_Document) range (Requirement))

Use of the SE Ontology

The SE Ontology can be used to support requirement management, trackability and traceability. By mapping concepts of the SE Ontology to a project requirements specification, the project requirement specification is more clear and precise towarding to structured and formal representation. In each requirements engineering phase, analysts are based on requirement domain knowledge described in the SE Ontology and have a shared understanding of the problem domain.

3.2 Design

Issues in multi-site software development

There are many notations used during design process to represent software design artifacts. When it comes to communicate over behavioral or structural descriptions of the design artifacts in multi-site software development environments it can be challenging. Different designers sometimes use different terminologies and it is often hard to overcome disagreement without face-to-face communication.

If a component-based approach is adopted, it is wise to find existing components to avoid re-working and it saves cost and can improve system quality in general. Most current repositories provide text search i.e. looking for keyword or plain text which will give low precision due to homonyms and low recall due to synonyms [4].

Description of the SE Ontology

The SE Ontology represents design knowledge formalism including architecture design, design notations, and design methods. For example it captures UML class diagram as following:

- Class ≡ ∃hasAttribute.Attribute ∪ ∃hasOperation.Operation ∪ ∃hasRelationship.Relationship
- Generalisation ⊆ Relationship
- ObjectProperty (relatingClass domain(Generalisation) range (Class)
- ObjectProperty (relatedClass domain(Generalisation) range (Class)

Use of the SE Ontology

With the SE Ontology, project design is attached to its semantic or main concept. This will reduce misunderstanding and eliminate ambiguity over the project design. Also the necessary details and relevant information can be retrieved automatically. For example in a project we identify the term 'customer' is a class. In the SE Ontology, concept class has its semantic of containing attributes, operations, and relationships holding among other classes. Thus details of class customer can be automaically and dynamically presented. The drawing out details facilitates others' greater understanding of the content. In traditional approaches project information normally resides isolated to its domain knowledge or background knowledge assuming everyone will understand the same which is not always happened especially in muli-site environment.

With project information attached to its concept, concept semantical search is more powerful and more convenient than simply plain syntactical keyword based search.

3.3 Construction

Issues in multi-site software development

In software construction, if object oriented approach is used some methods are called several times. There is currently no support in providing sequence of method calls [5]. This makes it even harder in multi-site environment if ones need to quickly learn the code. Another issue on software construction faced in both single- or multi-site software developments is a huge number of libraries. The documentation is vital for this issue.

Description of the SE Ontology

With interrelations of method calls in knowledge formalised in the SE Ontology, sequence of method calls can be generated i.e.

- Method_Calls ≡ ∃hasRelatingMethod.Method ∪ ∃hasRelatedMethod.Method
- Method_Sequence ≡ ∀ Relating_Method.Method ∪ Related_Method.Method

The SE Ontology captures code documentation which will be useful for maintenance of software systems. Below are some examples:

- DatatypeProperty (hasCodingAgreement domain (Code_Document) range (xsd: string))
- ObjectProperty (implementor domain(Code_Document) range(Software_Engineer))

Use of the SE Ontology

It is beneficial to remote teams when sequence of method calls can be generated. This also helps identify location of methods for coding support. With unified representation for code documentation in the SE Ontology, it enables easy cross references.

3.4 Testing

Issues in multi-site software development

Testing is a crucial part to ensure software quality [6] in multi-site software development. Creation of test cases is time consuming task with no direct business value [5].

Description of the SE Ontology

The SEOntology captures testing knowledge including test cases. Below are some examples:

- ObjectProperty(containAcivities domain (Test-case) range (Test_Log))
- ObjectProperty(containAcivities domain (Test-case) range (Defect_Tracking))
- ObjectProperty(hasObjectives domain (Test-case) range (Testing_Objectives))

Use of the SE Ontology

Test cases can then be generated given that it encodes the domain knowledge of software testing. Additionally in bug tracking system sometimes it requires finding classes that are related to a class which has bug. Possibly those classes may have bug as well. We can reason by taking following axiom:

- Bug_Related_Classes ≡ Bug_Relating_Classes ∪ ∀Relationship.Related_Class

4. Conclusion

There exist both advantages and disadvantages in deploying multi-site software development. We presented some issues and provided some solution into it by employing the SE Ontology. We demonstrated how the SE Ontology could help. Even though the SE Ontology requires modeling effort and endeavor of instances populating however we believe it pay off by having clear and unambiguous communication which is important in multi-site software development environments.

5. References

[1] P. Wongthongtham, A methodology for multi-site distributed software development, PhD thesis, Curtin University of Technology, Australia, 2006.

[2] P. Wongthongtham, E. Chang, T.S. Dillon, I. Sommerville, "Development of a Software Engineering Ontology for Multi-site Software Development", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, USA, 2008.

[3] I. Horrocks, P.F. Patel-Schneider, D.L. McGuinness, C.A. Welty, "OWL: a description logic based ontology language for the semantic web", The Description Logic Handbook: Theory, Implementation and Applications, ed. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. Cambridge University Press, 2nd edition, August 2007, pp. 458-486.

[4] A. Mili, R. Mili, R.T. Mittermeir, "A survey of software reuse libraries", Annual of software engineering, vol. 5, 1998, pp. 349-414

[5] H. Happel, S. Seedorf, "Applications of ontologies in software engineering", International Workshop on Semantic Web Enabled Software Engineering SWESE'06, Athens, USA, November, 2006

[6] A. Abran, J.W. Moore, P. Bourque, R. Dupuis, "Guide to the software engineering body of knowledge", 2004