The Evaluation of ontological representations of the SWEBOK as a revision tool

Miguel-Ángel Sicilia, Juan-José Cuadrado, Elena García, Daniel Rodríguez[2] and José R. Hilera
[1]Computer Science Department,
University of Alcalá, Spain
[2]University of Reading, UK
{msicilia, jjcg, elena.garciab}@uah.es, d.rodriguez-garcia@reading.ac.uk

Abstract

The SWEBOK represents an important milestone in reaching a broad agreement on the contents of the Software Engineering discipline. Formal ontologies thus become a tool to represent such agreement in a logics-based framework for a number of applications. In this paper, the use of common ontological criteria in the <Onto-SWEBOK> project is described as a useful assessment tool. The use of such concepts and a disciplined approach to representing terms and relations has resulted in a tentative structured revision procedure for SWEBOK material that could be used as a technique for improving or restructuring definitions. The main elements of the technique are described, along with illustrative examples of its potential application as a revision tool.

1. Introduction

The 2004 Guide to the Software Engineering Body of Knowledge (SWEBOK) is a significant milestone in reaching a broad agreement on the content of the software engineering discipline. Even though the project is not explicitly targeted at providing a common Software Engineering vocabulary, the usage of a clear and well-defined terminology is of paramount importance to eliminate ambiguities in any description of what constitutes the essential knowledge of the discipline. Natural language prose is useful for an efficient communication, but some applications require a higher level of formality of definition. For example, the cataloguing of learning resources or the mapping of vocabularies from different information sources require precise definitions, or at least significant *characterizations* that help in deciding which terms to use in practical situations. The IEEE Std 610.12-1990, labeled "IEEE Standard Glossary of Software Engineering Terminology" is a well-known attempt to provide precise characterizations of the main terms in the field. Nonetheless, it still fails in providing a clear demarcation for each of the concepts. For example, *software development lifecycle* is defined as "The period of time that begins with the decision to develop a software product and ends when the software is delivered". While this is understandable for human readers, ontologically it identifies lifecycles with `TimeInterval`s as defined in *OpenCyc* (the open source version of the *Cyc* knowledge base [6]). Considering this would be a common ontological error, since, for example, this definition would result in that comparing lifecycles would become a problem of comparing time intervals, i.e. the definition does not capture the essence of what a software lifecycle is. The concept of `Action` in *OpenCyc* captures much better the core characteristics of the concept. As defined in *OpenCyc*, `Events` (the constituents of actions) "should not be confused with instances of `TimeInterval`. The temporal bounds of events are delineated by time intervals, but in contrast to events time intervals have no spatial extent."

These kinds of ontological considerations require a disciplined approach to representing the SWEBOK. The <Onto-SWEBOK> project aims at engineering ontologies consistent with SWEBOK descriptions, as a framework for a number of concrete practical objectives. These objectives include the creation of ontology-based metadata records for learning objects [2], the provision of explicit "integration points" with other existing commonsense ontologies like *OpenCyc*, and the formal description of common Software Engineering frameworks like the Unified Process in terms of more generic conceptualizations, as a means to compare them. The <Onto-SWEBOK> method for ontology engineering combines standard practices [4] carried out in working teams with a literature-based approach [3] that helps in formally documenting and justifying the decisions made in connection with the text of the SWEBOK 2004. The ontology engineering work carried out to date includes the general terms found in all the Knowledge Areas and a complete study of the Requirements, Software Engineering Process, Software Engineering Management and Software Configuration Management. Practical problems found in the course of the engineering process have raised the need for a formal evaluation framework of ontology engineering decisions. The *OntoClean* [1, 5] method and general ontological considerations have been used for that purpose. The technique has proved useful in revealing some common problems in the process, and helped in a more rigorous inquiry of the kind of entities that were being modeled.

This has resulted in a technique that can be applied systematically to the SWEBOK text for ontology creation and review. Such technique could be used as a structured revision procedure for SWEBOK descriptions, useful in revealing ambiguities or excessively shallow definitions. In this paper, a sketch of such a revision technique is described and proposed as a formal evaluation tool for SWEBOK reference material.

The rest of this paper is structured as follows. In Section 2, the overall assessments method inside the <Onto-SWEBOK> project are described, illustrating some of the relevant problem types identified. Then, a revision procedure for SWEBOK material based on those evaluations is described. Finally, conclusions and possible directions for further work are provided in Section 4.

2. Engineering and Assessing SWEBOK-based ontologies

The SWEBOK provides a particular kind of narrative style in which terms are usually first introduced informally at the beginning of each Chapter, and then they are further explained in the rest. In addition, it uses citations to well-known reports, book and papers as a way to give external coherence to the concepts explained. In consequence, the SWEBOK can not simply be translated in a paragraph basis to ontological structures, since its definitions are not formal and require a degree of human interpretation and recurring to the literature explicitly cited. This is in fact a form of *literature-based* approach to description [3] that must be preserved in the crafting of the associated ontology. In addition, many terms and concepts explained appear several times thorough the book, which raises the need for consistency checks between parts of the book, with cross-referencing techniques like those used by Wille et al. [8].

The role of methodology for the engineering of the SWEBOK ontology has been emphasized elsewhere [7]. Nonetheless, here we are concerned with basic ontological assumptions and organizational principles that should be accounted for at the very beginning of the ontology engineering process. These principles serve as review criteria for the Guide in the sense that they provide a minimal classificatory framework to frame existing, modified and future concepts and relations.

The first important consideration is that Software Engineering should be considered as an artifact creation discipline. This leads to an important difference between *artifacts* and the reality they represent. Artifacts are in terms of *OpenCyc* information bearing things (`IBT`s) that are either components of the final system or *maps* oriented to the development of these components (e.g. diagrams, specifications). This is an important classification criterion that should be tested first. Two important issues must be considered with regards to this:

- `Artifact`s are distinct from the entities they represent, i.e., a *Requirement* is different form a *StatedRequirement*, being the latter a tangible artifact. Ontologically, the latter may not reflect perfectly the former, and this is an important fact to be recorded in actual processes, and not a mere accident. Artifacts have a clear mereology depending on the kind of product (e.g. document, source code, etc), so that their unity conditions [1] can be clearly stated.
- The software configuration as a support process in Software Engineering can be considered as an essential aspect of the discipline. This leads to the fact that every Artifact under such control is provided with an *identity* condition, i.e. the configuration identification.

Activities in Software Engineering are of a diverse nature. Nonetheless, all of them can be considered as `PurposefulAction`s (following *OpenCyc* terms), since they are carried out by `IntelligentAgent`s. These agents are humans or systems capable of “knowing”, and they can also be groups of them. The concept of intelligent agent covers the role of the developer, and all the variety of its *roles*. Actions are sequences of `Event`s and both concepts provide enough flexibility to model the wide diversity of concepts that relate to activities in the SWEBOK, including processes, technical procedures, steps or phases and so on. In any case, the actual activities carried out must be clearly distinguished from their specifications, which are rules that describe them. Activities are an important part of the SWEBOK, and each of them should be categorized from its early recognition in several dimensions, including: (a) kind of action and constituent actions, (b) artifacts created, modified or used, (c) agent roles involved, and (d) general rules prescribed. This is similar to process modeling, but differs in that it attempts to model both reality and *possibilia*, so that the criterion for inclusion is describing objects actually existing or specifications that bear a prescriptive (or eventually comparative) interest.

Other important ontological commitment is the representation of prescriptive knowledge. A concept of Method is available in *OpenCyc* as a way to relate actions required to accomplish other actions, but this is not rich enough to reflect all the varieties of prescriptive knowledge that appear in the SWEBOK. A concept of *Rule* can be used to subsume all of them, but specializations would require more detailed schemes. Rules may be inconsistent or contradictory with others, reflecting different theoretical or pragmatic viewpoints on ordering actions, ways of arranging or structuring artifacts or ways of producing them. This is an inherent characteristic that would require a separation of viewpoints in the final ontology. Even though the SWEBOK is intended to cover "accepted knowledge", supposedly consistent, actual instances or concrete cases may not have such coherence.

3. Ontological evaluations as a revision procedure for SWEBOK

The revision procedure that follows from the above mentioned assessment elements has been synthesized in a number of steps that are depicted in Figure 1.

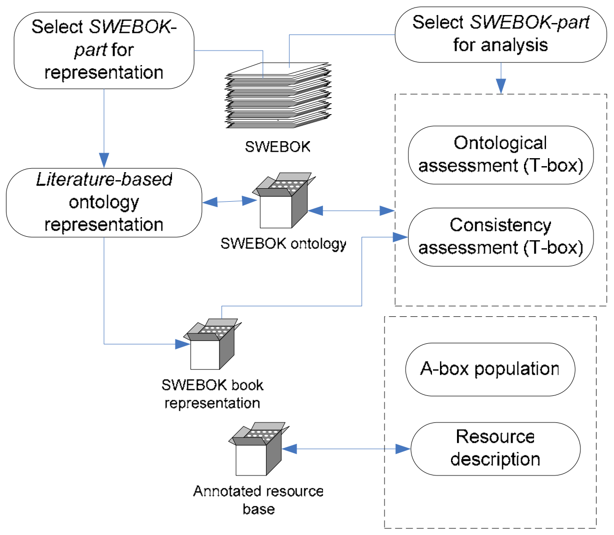
Select *SWEBOK-part* for representation
Select *SWEBOK-part* for analysis
SWEBOK
Ontological assessment (T-box)
Literature-based ontology representation
SWEBOK ontology
Consistency assessment (T-box)
A-box population
SWEBOK book representation
Resource description
Annotated resource base

Figure 1. Main elements of the ontology creation approach.

The procedure begins with a selection of the SWEBOK Guide as a target. Given the conceptual density of the writing of the Guide, small chunks no longer than a few pages are better suited for small ontology engineering teams with a concentrated effort schedule. The concepts in the "upper" part of the SWEBOK, as those described in the above section, serve as template for the rest of the process. Once the selection is made, two alternatives are possible:

- Ontology *representation*. Terms and relations are identified and represented in description logics (e.g. using the Protègè tool) by cross-referencing the terms in the chunk. The links with a large ontology must be documented or made explicit inside the ontology. Here both the ontology of SWEBOK and the representation of the Guide book are represented, and links are provided from terms in the former to instances on the latter. Principles and analysis techniques like [1] should be taken into account in this procedure.
- Ontology *analysis*. This procedure is comprised by a part in which the ontology terms are assessed, and other in which the links to the book representations are used for cross-analysis as in [8]. Both parts operate on the T-box of the ontology. In addition, a case-based assessment is used as a complement. This consists on populating the A-box of the ontology with concrete cases of the terms and relations, using real-world processes, models and artifacts. Moreover, learning resources are annotated with instances in the ontology (including reified terms as described in [3]) to provide a more intuitive understanding for future revisions.

It should be noted that analysis is not only concerned with the use of reasoners or consistency checkers but with the assessment of ontological decisions. The resulting structure enables different "external" reviewer roles (in addition to the engineers responsible for the process): (a) *software engineering experts*, that will use ontology-seeking interfaces to assess terms and instances, (b) *educators* and learning designers, that will browse annotated learning resources, and (c) *ontology experts*, navigating the whole structure and its connections to other ontologies.

These procedures enable the iterative refinement of the ontology. A useful but expensive technique following this procedure is that of assigning the same SWEBOK Guide chunk to two separate teams, providing a basis for comparison and critique on ontological decisions.

4. Conclusions and Future Research Directions

The engineering of an ontology based on the Guide to the SWEBOK is able to providing insights in the revision of the Guide itself. We have described some core ontological commitments adopted in the <*Onto-*

SWEBOK> project, and a review method based on that experience has been described.

Ongoing work in the ontology of SWEBOK should ideally provide feedback for the Guide itself. Future work should continue the current ontology development efforts combined with ontology-based seeking interfaces [9]. This would also help in the ongoing revision of the ontology and the Guide.

5. References

[1] Guarino, N. and Welty, C. Evaluating ontological decisions with OntoClean. *Communications of the ACM* 45(2), 61-65.
[2] Polsani, P. R. (2002). Use and abuse of reusable learning objects. *Journal of Digital Information*, 3(4).
[3] Sicilia, M.A., García, E., Aedo, I. and Díaz, P. (2003). A literature-based approach to annotation and browsing of Web resources. *Information Research* 8(2).
[4] Ushold M. & Gruninger M., (1996), Ontologies: Principles, methods and applications, in *The Knowledge Engineering Review*, 11 (2): 93 – 155.
[5] Welty, C. and Guarino, N. Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering* 39(1), 2001, pp. 51-74.
[6] D.B. Lenat, Cyc: A Large-Scale Investment in Knowledge Infrastructure'. *Communications of the ACM* vol. 38, no. 11, pp. 33--38, 1995.
[7] Mendes, O., Abran, A., Software Engineering Ontology: A Development Methodology, in *Metrics News* ,vol. 9 , 2004 , pp. 68-76 .
[8] Wille, C., Abran, A., Desharnais, J.M., Dumke, R.R. (2003). The quality concepts and subconcepts in SWEBOK: An ontology challenge, in International Workshop on Software Measurement (IWSM), Montreal, pp. 113--130.
[9] García, E. & Sicilia, M.A. (2003). User Interface Tactics in Ontology-Based Information Seeking. Psychology e-journal 1(3):243-256.