Ontologies of Software Artifacts and Activities: Resource Annotation and Application to Learning Technologies

Miguel–Angel Sicilia[1], Juan-José Cuadrado[1] and Daniel Rodríguez[2]
[1]University of Alcalá, Madrid (Spain)
{msicilia,jjcg}@uah.es
[2] University of Reading, UK
d.rodriguezgarcia@reading.ac.uk

Abstract

The emerging consensus on the boundaries and main elements of the Software Engineering (SE) discipline represents an opportunity for the engineering of shared conceptualizations that may serve both to design automated tools and tasks that help in diverse phases and aspects of the software process, and also to annotate learning–oriented resources. Formal ontologies provide an appropriate logics–based framework for such conceptual models. This paper describes the main ontological commitments that underlie the `Onto-SWEBOK` *project, focusing on how SE artifacts and activities can be represented. The paper also discusses how semantic annotations can be provided for learning resources oriented to the initial and continuing education on the discipline, which enables the reuse of such resources in diverse learning designs.*

1. Introduction

The 2004 Guide to the *Software Engineering Body of Knowledge* (SWEBOK[1]) is a significant milestone in reaching a broad agreement on the content of the Software Engineering discipline. The SWEBOK is aimed at developing a consensus in what constitutes "validated", rational and scientific Software Engineering (SE) knowledge. As such, it represents a concrete, shared view of the discipline that rests on a collection of interrelated concepts. However, the current form of the SWEBOK uses natural language text and technical narrative to describe the main elements and boundaries of the discipline. While this is appropriate for human communication, a formalized version of the text is required for the development of diverse tools that use computational semantics [15] to manage, search and trace diverse kind of SE resources or representations.

[1]http://www.swebok.org

Formal ontologies are engineered artifacts aimed at representing a shared, consensual conceptualization of a given domain [9]. Description logics [2] and extensions are the underlying representation framework for ontologies as they are used in the so–called Semantic Web applications [3]. In consequence, the engineering of a formal ontology representing the SWEBOK would enable knowledge reuse and a standardized model for the cataloguing of SE artifacts. In addition, the process of ontology engineering could be used as a tool for revision of the SWEBOK, considering that ontological analysis methods [19, 4] are aimed at clarifying the conceptual structures and properties of a given domain.

Previous work have addressed some aspects of engineering formal representations of SE elements. Falbo et al [7] reports on the use of shared conceptualizations for integrated tool development, and Althoff et al [1] describe an architecture oriented to reuse of experience in SE that uses ontologies as the underlying formalism. Deridder et al have used ontologies for the specific purpose of linking artifacts at several phases of the development process [5]. Nonetheless, the knowledge representation underlying these systems are conceptual models aimed at enabling concrete functionality, and not oriented to describing the main ontological commitments [19] of the discipline. The SWEBOK project opens new possibilities to ontology engineering in the field of SE, since it represents a shared consensus on the contents of the discipline and provide pointers to relevant literature on each of its concepts, which are two important elements in ontology engineering [9, 17]. Existing research has provided insight in ontological representation problems on some of the areas of the SWEBOK [20] and on the techniques used for ontological representation [12]. Nevertheless, work is still needed in the analysis of the main ontological commitments of SE as a discipline, and the representation of both its "upper" concepts and the resources that refer to them.

In this paper, the use of formal ontologies as a tool to

annotate software *entities* and associated *resources* is described. Here the term "entity" is used to refer to existing elements (be them physical or temporal), and "resource" is used to refer to information bearing things that can be related to concept describing SE entities. The concrete approach described is based on a layered ontological structure that facilitates meta–description (even with languages that lack such logical facility), and it is flexible enough to accommodate a diversity of applications that deal with entities or resources. The use of annotations connected to the ontology of SWEBOK as a means to devise reusable learning materials is provided as an example of the latter.

The rest of this paper is structured as follows. Section 2 provides the core ontological elements of the ontology of SWEBOK developed as part of the `Onto-SWEBOK` project. Then, Section 3 describes how actual entities and any kind of discourse about them can be represented. Then, Section 4 sketches the main elements of the integration of the ontology described with existing learning technology standards. Finally, conclusions and future research directions are provided in Section 5.

2. Ontological representations of SWEBOK Activities and Artifacts

Engineering sciences are mainly concerned with *design*, e.g. as described by Mitcham "designing is a special kind of activity that so far has found almost no place in what is known as the "philosophy of action", that is, the reflective analysis of the distinctive characteristics of human behavior" [13]. In consequence, it deals with purposeful *activities* that use and create *artifacts*. Then, the core classes subsuming the rest of the more concrete concepts and relationships in the ontology of SE should somewhat deal with *artifacts* created by *agents* as a result of codified *activities* guided by *rules*. All these elements can be found in current large commonsense knowledge bases as *OpenCyc* (the open source version of *Cyc* [11]). The following list provides the mapping of these elements (OpenCyc terms are prefixed by "`oc_`")

- Developers in a generic sense, including "intelligent" systems that aid in the process can be considered as instances of `oc_IntelligentAgent`, "agents capable of knowing and acting, and of employing their knowledge in their actions. An intelligent agent `oc_knowsAbout` certain things, and its `oc_beliefs` (and possibly `oc_goals`) concerning those things may influence its actions. [...] an intelligent agent might be a single individual or might consist of a group of individual agents (see `oc_MultiIndividualAgent`)."

- Activities in *OpenCyc* can be regarded as `oc_Action` "The collection of `oc_Events` that are carried out by some "doer" (see `oc_doneBy`). Instances of Action include any event in which one or more actors effect some change in the (tangible or intangible) state of the world, typically by an expenditure of effort or energy." An `oc_Event` is in turn "a dynamic situation in which the state of the world changes; each instance is something one would say 'happens'." Moreover, engineering activities are in fact `oc_PurposefulActions`, "Each instance of PurposefulAction is an action consciously, volitionally, and purposefully done by at least one actor". In addition, the notions of *design* and *designing* actions in *OpenCyc* extend these definitions to activities oriented to produce `oc_Specifications` which are a prominent category of artifacts in the domain of SE.

- An `oc_Artifacts` is "an at least partially tangible thing which was intentionally created by an `oc_Agent` (or a group of Agents working together) to serve some purpose or perform some function."

- Rules in a general sense include in the SE domain prescribed sequences of actions, desired characteristics of artifacts and general organizational constraints. These rules are of diverse nature, and in *OpenCyc* they would be regarded as instances of `oc_SupposedToBeMicrotheory`, which group together assertions that describe how things are "supposed to be" according to some source.

The use of the just described conceptual framework to the descriptions of each SWEBOK Knowledge Area (KA) in the first chapter of the book resulted in the tentative identification of elements as those provided in Table 2. The method for elicitation was simply that of contrasting literal definitions in the SWEBOK Guide to the categories described above.

The examination of the "Requirements" KA is illustrative of some of the main ontological commitments of the activity–artifact framework:

i A clear distinction should be drawn between *Artifacts* and "real world" entities. From a pragmatic ontological perspective, "real things" are not necessarily tangible, in the sense that they are not made of matter, and they are not definitely created by any kind of SE *IntelligentAgent*. A relevant example is a *user need*. Nonetheless, in many cases we are only concerned with the representations, since they are the only information we have about the "real" entities, e.g. the *system bound* is not a physical frontier but the delimitation of the system as represented in stakeholder's desires.

KA	*Action*	*Artifact*	*Other*
Requirements	(Requirements) elicitation, analysis, specification, validation and management	Requirements document	(System and software)requirement, requirements engineer (agent), requirement source, system bound, requirement conflict
Design	Change management activity, quality analysis	requirement-back-link (trace), architecture blueprint	Persistence, event, design pattern, object-oriented method (rules), quality attribute, design notation
Construction	Coding, testing	source code file, unit test description	Complexity reduction, diversity anticipation (rules)
Testing	Verification activity	Test case, test-related measure	Defect, verification technique (rule)
Maintenance	Enhancement process	Change request, incident report	Anomaly
Configuration Management	Software configuration identification, software configuration status accounting	Configuration, version record	Identification scheme
SE Management	Organizational management, scope definition, planning	Project plan, function point count	Gantt notation, tracking assessment criterion
Process	Qualitative process analysis, process implementation	Process model	Software Lifecycle
Quality	Formal review	Review report	Quality attribute

Table 1. Main actions and artifacts (produced by SE actions) in the KA

ii Parts of artifacts are information bearing things (`oc_IBT`) that are representations of other elements. For example, a use case diagram is an artifact that includes representations of requirements and their intrinsic relationships.

iii Actual actions should be differentiated from methods or other prescriptions of sequences of actions. Methods in *OpenCyc* can be represented through `oc_methodForAction` predicates connecting an action with an action sequence that describes an appropriate method for carrying out the action. This way, requirement elicitation techniques can be separated from the the actual events that conform the process followed in a specific project.

iv Rules should be separated in *microtheories* that are only constrained to be internally consistent. Each microtheory may represent a different position on methodological issues, e.g. "extreme" versus "traditional" development, reflecting the changing nature of many SE approaches.

Commitment [i] above is useful for the encoding or rules implicit in statements like "examining the requirements document to ensure that it defines the right system (i.e. the system that the user expects)". This is a clear case in which the distinction becomes important from a methodological perspective. In addition, artifacts are rigid entities with a clear mereology of parts, according to *OntoClean* definitions [19], while real world entities are not necessarily characterized that way.

The action-event dichotomy described above is sufficient for the representation of temporal events that represent a change in the engineering context. However, other The IEEE Std 610.12 - 1990 document defines *process* as "a sequence of steps performed for a given purpose", *procedure* as "a course of action to be taken to perform a given task", a *technique* as "technical and managerial ***procedures*** that aid in the evaluation and improvement of the software development process", and method standard as "a standard that describes the characteristics of the orderly ***process*** or ***procedure*** used in the engineering of a product or performing a service". These definitions have a degree of ambiguity, so that the generic *method* term will be used to subsume the other three concepts.

3. Annotating Software Engineering Artifacts and Activity Descriptions

The `Artifact` and `Action` terms are the subsumers of SE results and activities respectively. `EngineeredArtifacts` are by necessity `producedBy` some `EngineeringAction`. This is a form of traceability by which the objectives of activities become explicit.

A layered approach to the ontology is depicted in Figure

1, with upper concepts representing those that can be found in knowledge bases like *OpenCyc*.

Three parts of the ontology should be clearly differentiated. The *description* part groups the *T-box* and *A-box* (in terms of description logics [2]) that characterize artifacts and activities as created and enacted by actual SE practice. These could be used for process representation as in [7], and constitute by themselves a form of ontology of actual software configuration items, and a comprehensive record of activities that could be used for tracking in project management. On the contrary, the *prescriptive* part deals with a different aspect of reality, which comprises the approaches or *rules* to concrete practical activities that are "commonly accepted" as considered in the SWEBOK. Even though a degree of consensus exists for them, nothing prevents the possibility of conflict and inconsistency. For example, two process models may differ in the importance they give to some artifacts or activities ("extreme" and agile approaches as compared with classical ones are a relevant example). In addition, several competing techniques may exist for some actions from which there is not a strong evidence about which one is the "right" one. The *Cyc* [11] concept of "microtheory" allows the grouping of different methodological or pragmatical standpoints, providing the flexibility required for an evolving discipline as SE. In this context, methods in a general sense are the rules that tell something about how to carry out sequences of `Actions`.

An additional aspect of the SWEBOK as a knowledge base is the representation of *resources*. Here the term resource is assimilated to that of `IBTs` that provide information about the rest of the elements of the ontology (i.e. they are essentially *dependent* on them [19]). Two classes of resources are distinguished. The literature on SE is the compendium of `IBTs` that describe elements of the discipline, and as such, links from ontology terms and instances are provided through a reification technique as that described in [17]. Resources in general are any `IBT` or `ConceptualWork` that predicates on elements of the SE ontology. Both parts enable the straightforward building of concept browsers that use ontology–based seeking techniques [8] to locate resources or to go from resources to concepts or instances. This cross-linking provides the infrastructure for technique of analysis as the one described by Wille et al. [20].

The concept of "annotation" in ontology–based representations does not require a fixed data schema prescribing some "fields" that should be filled for item depending on its type. On the contrary, any axiom or property inside the ontology can be considered a predication that describes something useful from a broad view of semantics [15]. In consequence, the ontological descriptions of all the elements described can be considered as annotations in themselves. For example, the representation of a requirements document as an instance of the corresponding class would have as annotations all other predicates inside the ontology that refer to it. If only a part of them would be desired to be used, a simple mechanism providing "super–slots" could be applied, i.e. high-level predicates or relations that subsume the ones that are considered. For example, a property `trace-relation` could be used as a subsumer of the variety of relations that may be interpreted as carriers of information about traceability as `derivedFrom` between artifacts or `scheduledIn` between activities (e.g. activities that are scheduled as the result of a meeting discussion).

4. Integrating Standardized Learning Technologies with the Ontology of SWEBOK

Current approaches to Web–based learning are based on the concept of *learning object*, for which several definitions have been proposed. Reusability is considered to be an essential characteristic of the concept of learning object as the central notion for modern digital learning content design. For example, Polsani [14] includes reuse in his definition of learning object as "an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts", and Wiley [21] also mentions the term in his learning object definition "any digital resource that can be reused to support learning". Existing work has dealt with the integration of that concept in *OpenCyc* [18]. Learning objects by definition are described by metadata records. IEEE LOM [10] is a relevant standard that determines a collection of metadata fields for learning objects.

Given the above ontological structure, `LearningObject` is a defined concept that represents anti–rigid entities subsumed by `IBT` that become learning resources by virtue of the declaration of their possible educational usages. It is of special relevance to consider that every digital `Artifact` represented is by itself a learning resource, at least as an exemplar of an element of the discipline. The annotations provided in the sense described in the previous section could be used to decide on their use in learning activities. But this view is loose with respect to automated reuse, since the annotations are *descriptive* rather than *normative* in the sense described by Sánchez-Alonso and Sicilia [16]. This calls for an additional layer in the ontology that explicitly targets learning objectives. The following provides an illustration on such additional representation based targeting examples relevant to the *IEEE/ACM Computing Curricula 2001* (CC2001).

"Software requirements and specifications" is a core topic in CC2001, and the the differentiation between functional and non-functional requirements is one of the skills to be acquired. A learning object covering such element could provide some brief description of the topic (thus

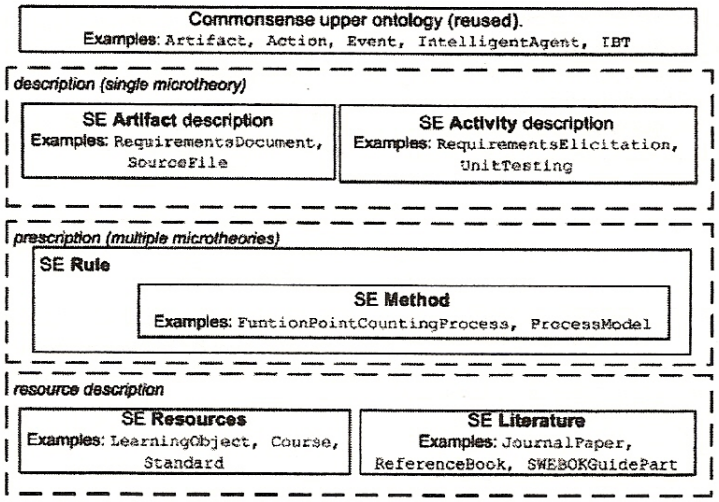

Figure 1. Overall structure of the ontology

being self-standing and of lower granularity as recommended in [14]) and then provide an adaptive part which takes from an ontological *A-box* (which we assume to have the representation of several software projects) an instance of `Functional-Requirement` and another of `NonFunctional-Requirement` from the same `Project`. The approval status of the requirements in the project would act as a filter for selecting only "correct" requisites. In addition to this, the non-functional requirement could be restricted to be connected to some `QualityAttribute` instance (e.g. `efficiency` or `usability`) to highlight such relation, while a trace to derived artifacts could be showed for the functional one (reaching the final source code if available). Note that this kind of designs for learning *reuse* actual SE results.

Following the example, the learning object thus designed would provide the following normative usage conditions expressed in terms of a *contract* [16]:

- The learner should know some basic material, e.g. the `lrn.knows(con-req-def)` precondition is required, stating that the concept of software requisite should be previously mastered by the learner.
- The postcondition should state the (expected) increase in understanding of what functional and non-functional means for requisites. This correspond with the intention of the *(9.1.) Purpose* metadata element in the IEEE LOM standard [10].

In addition, the kind of resource (LOM 5.2. element) should be set to "illustration" as a specific kind of resource that exemplifies through project data. These and other metadata elements enable the dynamic selection of learning objects for concrete situations, including informal training. For example, the above learning object could be selected automatically by a CASE tool as a help item in a requirement tool, or it may be selected as part of a degree on Computer Science covering the contents of the CC2001.

Since normative definitions as the above do not preclude the same object to be described for a different usage (in the sense of Downes' resource profiles [6]), the labor of learning design in this context would benefit of an unprecedented level of reuse, including referencing to available and relevant actual professional practice results.

5. Conclusions and Future Research Directions

A core set of concepts borrowed from *OpenCyc* has been used to delineate the main ontological commitments of an approach to the ontology of SE as a disciplined artifact–creating discipline. The descriptive versus prescriptive aspects have been differentiated, and the dichotomy between representations in artifacts as documents and the actual "reality" has been described as a way to state rules of a general kind. This basic structure may be extended or modified in further work, but in any case it serves as a foundation for research on the topic.

Such kinds of discipline–oriented ontologies provide a clear value in their applicability to facilitate education and training, since they can be used as tools for metadata annotation. This paper has discussed this with reference to describing *learning objects*.

Ongoing work in the Onto-SWEBOK project should address many ontological definitions that are not currently explicit in the Guide to the SWEBOK. The use of existing large conceptualizations and common ontological analysis techniques will be used to eventually come up with a ontology of SE based on notions of engineering as technology–creating endeavors.

References

[1] Althoff, K.-D., Birk, A., Hartkopf, S., Müller, W., Nick, M., Surmann, D. and Tautz, C. (2000). Systematic Population, Utilization, and Maintenance of a Repository for Comprehensive Reuse. In G. Ruhe and F. Bomarius (eds.), *Learning Software Organizations - Methodology and Applications*, Springer Verlag, Lecture Notes in Computer Science, LNCS 1756, 25-50

[2] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.). (2003). *The Description Logic Handbook. Theory, Implementation and Applications*, Cambridge.

[3] Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34-43.

[4] Brachman, R. (1983). What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer* 16(10), 30-36.

[5] Deridder, D., Wouters, B., Lybaert, W. (2000). The Use of an Ontology to Support a Coupling between Software Models and Implementation. In *Proceedings of the International Workshop on Model Engineering*, 14th European Conference on Object-Oriented Programming (ECOOP).

[6] Downes, S. (2004). Resource Profiles. *Journal of Interactive Media in Education*, 2004 (5).

[7] Falbo, R.A., Natali, A. C. C., Mian, P. G., Bertollo, G., Ruy, F. B. (2003). ODE: Ontology-based software Development Environment. In *Proceedings of the "IX Congreso Argentino de Ciencias de la Computación"*, 1124-1135.

[8] Garcia, E. and Sicilia, M.A. (2003). User Interface Tactics in Ontology-Based Information Seeking . *Psychnology e-journal* 1(3), 243-256.

[9] Gruber T. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6), 907-928.

[10] IEEE LTSC (Learning Technology Standards Committee). (2002). *Learning Object Metadata (LOM)*. IEEE 1484.12.1-2002.

[11] Lenat, D.B. (1995). Cyc: A Large-Scale Investment in Knowledge Infrastructure". *Communications of the ACM* 38(11) 33–38.

[12] Mendes, O., Abran, A. (2004). Software Engineering Ontology: A Development Methodology. *Metrics News*, 9, 68-76.

[13] Mitcham, C. (1994). *Thinking through Technology: The Path between Engineering and Philosophy*. Chicago: University of Chicago Press, xi, 397.

[14] Polsani, P. R. (2003). Use and Abuse of Reusable Learning Objects. *Journal of Digital information*, 3(4).

[15] Seth, A. et al.(2005). Semantics for the Semantic Web: The Implicit, the Formal and the Powerful. *Intl. Journal on Semantic Web and Information Systems* 1(1), 1-18.

[16] Sánchez-Alonso, S. and Sicilia, M. A. (2005). Normative Specifications of Learning Objects and Learning Processes: Towards Higher Levels of Automation in Standardized e-Learning. *International Journal of Instructional Technology and Distance Learning*, 2(3), 3-12.

[17] Sicilia, M.A., Garcia, E., Aedo, I. and Diaz, P. (2003). A literature-based approach to annotation and browsing of Web resources. *Information Research* 8(2).

[18] Sicilia, M.A., García, E., Sánchez, S. and Rodríguez, E. (2004) Describing learning object types in ontological structures: towards specialized pedagogical selection. In *Proceedings of ED-MEDIA 2004 - World conference on educational multimedia, hypermedia and telecommunications*, 2093-2097.

[19] Welty, C. and Guarino, N. (2001) Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering* 39(1), 51-74.

[20] Wille, C., Abran, A., Desharnais, J.M., Dumke, R.R. (2003). The quality concepts and subconcepts in SWEBOK: An ontology challenge , in *Proceedings of the 2003 International Workshop on Software Measurement (IWSM)*, 113–130.

[21] Wiley, D. A. (2001). *The Instructional Use of Learning Objects*. Association for Educational Communications and Technology, Bloomington.