APPLYING THE UNIFIED PROCESS TO LARGE-SCALE ONTOLOGY BUILDING

Michele Missikoff, Roberto Navigli

Consiglio Nazionale delle Ricerche , Roma Italy, missikoff@iasi.cnr.it
Università di Roma "La Sapienza", Roma, Italy, navigli@di.uniroma1.it

Abstract: Ontologies are the backbone of the Semantic Web, a semantic-aware version of the World Wide Web. To the end of making available large-scale domain ontologies, effective and usable methodologies are needed to facilitate the process of Ontology Building. Many of the methods proposed so far only in part refer to well-known and widely used standards from other areas, like software engineering and knowledge representation. In this paper we present UPON, a methodology based on the adaptation to Ontology Building of the Unified Software Development Process developed by the Object Management Group. The approach provides interesting insights from both software engineering and ontology construction. A comparative evaluation with other methodologies, as well as the results of its adoption in the context of the European Interop Network of Excellence and Athena Integrated Project, are also discussed.

Keywords: knowledge engineering, software engineering, knowledge representation, knowledge acquisition, documents, domain analysis.

1. INTRODUCTION

Ontologies, i.e. semantic structures encoding concepts, relations and axioms for inference, are the backbone of the Semantic Web (Berners-Lee et al., 2001), a semantic-aware version of the World Wide Web. The availability of large-scale domain ontologies is a critical factor for achieving semantic services, and in particular interoperability among systems.

Unfortunately the community has not yet reached a de facto consensus on one or more standard methods for building large-scale ontologies. This seems to be a case where a field of AI could benefit from borrowing the main, well-established characteristics of a widely used software engineering process, the Unified Software Development Process (Jacobson et al., 1999).

In this paper, we present UPON, a novel approach to large-scale ontology building based on the Unified Process (UP). As a result, on one side, the adoption of the UP and the Unified Modeling Language (UML) makes ontology building an easier task for modellers familiar with these techniques. On the other side, we show how well each phase of ontology building fits in the UP, thus guiding the process through a number of consolidated steps aiming at ontology development.

The paper is organized as follows: section 2 discusses previous work in this area, in section 3 we present our approach to ontology building, in section 4 we provide a two-fold evaluation of UPON, the first by comparison with other methodologies, and the second in the context of a European Network of Excellence on interoperability, Interop[1], and the Athena[2] Integrated Project. Finally, in section 5 we provide conclusions and future work.

---

[1] "Interoperability Research for Networked Enterprises Applications and Software", Network of Excellence 508011, 6th European Union Framework Programme (FP) - http://www.interop-noe.org.

[2] "Advanced Technologies for Interoperability of Heterogeneous Networks and their Application", Integrated Project 507849, 6th EU FP.

2. RELATED WORK

Among the first and most cited contributors to ontology building, Gruber (1993) discusses some basic ontology design criteria (clarity, coherence, extendibility, minimal encoding bias and ontological commitment). Uschold and Gruninger (1995) provide a skeletal methodology based on the identification of purpose, the construction of the ontology, its evaluation and documentation. Gruninger and Fox (1995) point out the need of establishing requirements with the aid of competency questions.

These works introduce the main guidelines for building an ontology, constituting the basis for the subsequent proposals.

A complete framework for ontology development, *METHONTOLOGY*, is presented by Fernández et al. (1997). The ontology development process is composed by the following phases: specification, conceptualization, formalization, integration, implementation, maintenance. Its life cycle is based on evolving prototypes and specific techniques peculiar to each activity. Other activities, like control, quality assurance, knowledge acquisition, integration, evaluation and documentation are carried out simultaneously with the ontology development activities.

With a strong emphasis on knowledge maintenance and management, Sure et al. (2004) propose *On-To-Knowledge*, an ontology development process consisting of five main phases: feasibility study, kick-off, refinement, evaluation, application and evolution. Each phase consists of a number of sub-steps. The process is well detailed, including pre- and post-development phases.

Other approaches, often tied to industry or research projects, include the methods used for building *CyC*, *SENSUS*, and *KAKTUS* (OntoWeb deliverable, 2002). For an accurate description of ontology building methodologies the interested reader can refer to Corcho et al. (2003).

Unfortunately the ontology community has not yet reached a de facto consensus on the above mentioned methodologies paired with an intuitive modeling language. In the rest of the paper we propose, discuss and assess an approach to the problem.

3. UPON: UNIFIED PROCESS FOR ONTOLOGY BUILDING

In this section we present UPON (Unified Process for ONtology building), an incremental methodology for ontology building. The process we propose stems its characteristics from the Software Development Unified Process, one of the most widespread and accepted methods in the software engineering community, and uses the Unified Modeling Language (UML) to support the preparation of all the blueprints of the ontology project[3]. UML has been already shown to be suitable to this end (Guizzardi et al., 2002), confirming its nature of rich and extensible language.

What distinguishes the UP and UPON from the other processes, respectively for software and ontology engineering, is their *use-case driven*, *iterative and incremental* nature.

UPON is *use-case driven* in that it aims at producing an ontology with the purpose of serving its users, both humans and automated systems (e.g. semantic web services, intelligent agents, etc.). These interactions take place through *use cases* that drive the exploration of all aspects of the ontology.

The nature of the process is *iterative* because each iteration allows the designer to concentrate on part of the ontology being developed, but also *incremental*, since the ontology is more and more detailed and extended.

The process repeats over a series of cycles making up the life of the ontology. Following the UP, in UPON we have cycles, phases, iterations and workflows. Each cycle consists of four phases (*inception*, *elaboration*, *construction* and *transition*) and results in the release of a new version of the ontology. Each phase is further subdivided into iterations. During each iteration, five workflows (described in the next subsections) take place: *requirements*, *analysis*, *design*, *implementation* and *test*. Workflows and phases are orthogonal in that the contribution of each workflow to an iteration of a phase can be more or less significant: early phases are mostly concerned with establishing the requirements (identifying the domain, scoping the ontology, etc.), whereas later iterations result in additive increments that eventually bring to the final release of the ontology (Figure 1).

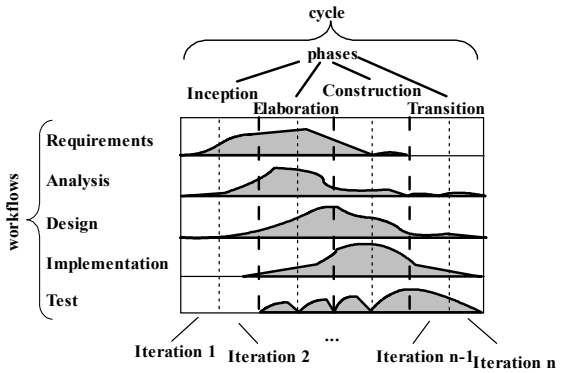

Fig. 1. An example of the workflow contribution to iterations of different phases in UPON (and UP).

In the following subsections each ontology workflow is described in detail.

3.1 The Requirements Workflow.

Requirements capture is the process of specifying the knowledge to be encoded in the ontology. The essential purpose of this workflow is to reach an agreement between the modellers, the knowledge engineers, and the final users (Jacobson et al., 1999).

[3] Notice that in the following we apply UPON to the construction of an ontology, rather than to the process of building ontologies, that is the blueprints refer to the ontology being built.

During the first meetings, ontology modellers and domain experts establish the guidelines for building the ontology by: (i) determining the domain of interest, (ii) defining the purpose and scope, and (iii) identifying the competency questions and the related use cases.

Determining the domain of interest. Delimiting the domain of interest is a fundamental step to be performed (Uschold and Gruninger, 1995), allowing to focus on the appropriate fragment of reality to be modelled. If the domain is huge, one or more sub-domains may also be determined.

Defining the purpose (or motivating scenario). The reason for a new ontology, its intended uses, and the kinds of users must be established. Common motivating scenarios provide a better understanding of the domain of interest and foster interoperability between systems and/or users.

Defining the scope. The scope is the extent of the ontology and consists of the identification of the most important concepts to be represented, their characteristics and granularity. Selecting a representation means making a set of ontological commitments, bringing some part of the domain into focus at the (required and expected) expense of blurring other parts. These ontological commitments are not incidental: they provide a guidance in deciding what aspects of the domain are relevant and what to ignore.
Following Guarino et al. (1994), the ontological commitment can be seen as "a mapping between a language and something which can be called an ontology". This allows one to preliminarly identify terms as expressions of ontology concepts.
Usually at this stage modellers have only a vague idea of the role each concept will play, i.e. the semantic interconnections, within the ontology. If necessary, they can annotate these ideas for further development during subsequent iterations.

Identifying the competency questions. Competency questions are questions an ontology must be able to answer (Gruninger and Fox, 1995). They are identified through interviews with domain experts, brainstorming, an analysis of the document base concerning the domain, etc. The questions do not generate ontological commitments, but are used during the test workflow to evaluate the ontological commitments that have been made.

Use-case identification and prioritization. We propose to take into account competency questions through use-case models. A *use-case model* serves as an agreement between the users (i.e., who requires the ontology) and the modellers, and contains a number of *use cases*. In the context of ontologies, use cases are simply paths of knowledge through the ontology to be followed for answering one or more competency questions. Although they are to be specified during the analysis and design workflows, it is necessary to *prioritize* and *package* (i.e. group) them during requirements. The result will help dictate which use cases the team should focus on during early iterations, and which ones can be postponed.
An example of use-case model applied to competency questions is reported in Figure 2.

Competency Questions:
Which actors integrate heterogeneous data?
Who acquires knowledge?
Use-case model:

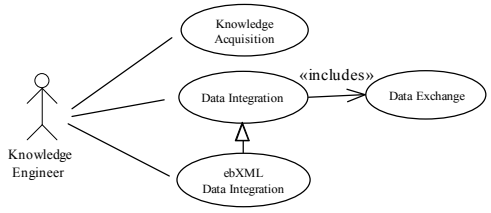

Fig. 2. Some competency questions and an excerpt of the use-case model.

The outcome of the requirements workflow is a first document including the results of the above steps. Figure 2 shows part of the resulting draft document developed in the context of the Interop Network of Excellence on interoperability. The document is extended and detailed during subsequent iterations, as the whole picture becomes clearer.

3.2 The Analysis Workflow.

Conceptual analysis consists of the refinement and structuring of the ontology requirements identified in previous section. The ontological commitments derived from the definition of scope are extended, by reusing existing resources and through concept refinement.
The description of this phase adheres to the view of *linguistic ontology* in which concepts, at least the lower and intermediate levels, are anchored to texts, i.e. they have a counterpart in natural language.

Considering reuse of existing resources. Reuse concerns internal legacy resources as well as external resources requiring possible refinements and extensions, like interviews, documents, standards (e.g. ebXML, RosettaNET, OAGIS, etc.), glossaries, thesauri, computational lexicons and available ontologies. Compared to building the ontology from scratch, populating part of it with existing resources can save a great deal of time and produces an ontology with higher interoperability features. Reuse always implies some kind of integration and adjustment (merging, mapping, reinterpretation, etc.).

Identification of relevant terms. During conceptual analysis, domain modellers identify the linguistic realization of the entities involved in the domain of interest starting from the work performed during the requirements capture. The result of this step is a plain list of terms. Notice that a domain term characterizes a domain concept in an implicit way. Making this knowledge explicit is an important effort to be accomplished partly during analysis (in the next step)

and partly in the design workflow. Table 1 shows part of the terminology captured in the Interop project, by analysing the document base.

Table 1 Plain terminology.

business entity	design phase	legacy system
business object	domain expert	resource
data integration	interoperability	software agent
deployment	knowledge source	…

Definition of concepts. Starting from the plain term list, unnamed relations can be identified implying some kind of conceptual correlation or interaction between the concepts evoked by terms in the list. This results in the *Class* (or *Concept*) *Responsibility Collaborator* (*CRC*) model, well-known in the software engineering area. This model is a collection of standard index cards, each divided into three sections: the concept *name*, its *responsibilities* (what an instance of that concept knows or does) and its *collaborators* (concepts it interacts with to fulfill its responsibilities). Each card can be further enriched with a definition in natural language and the identification of a top-level (or meta-level) "category" for the defined concept (e.g. *entity* for Business Entity, *process* for Data Integration, *actor* for Software Agent, etc.). These "categories" include the major ontological types (usually concepts either in the top ontology, resulting from various permutations of very basic facets, or constituting an established meta-ontology (Uschold and Gruninger, 1995; Missikoff and Taglino, 2002)). An example of extended CRC card is shown in Figure 3. Notice that the textual definition can be influenced by the concept responsibilities or vice-versa. Synonyms, i.e. terms expressing the same meaning (much as synsets in WordNet (Fellbaum, 1998)), are grouped into the same card. For instance, the terms *ontology construction* and *ontology building* convey the same meaning, so they can be grouped in order to represent a single concept.

Concept name: Knowledge Engineer <Actor>	
Responsibilities: interview experts acquire knowledge	**Collaborators**: Domain Expert Knowledge Source
Definition: A person who communicates with experts in order to acquire relevant knowledge.	

Fig. 3. Part of an extended CRC card for Knowledge Engineer.

The CRC model can be represented in a graphical form with a UML *collaboration diagram* (called *robustness*) where *entities*, *actors* and *processes* are identified with UML standard graphical symbols (refer to Figure 4 for an example).
According to the UP methodology, collaboration diagrams suggest that concepts can be grouped into packages, aiming at organizing the development in mini-projects at different iterations.
The outcome of this workflow is the analysis model, including the packages of concepts and collaboration diagrams. A portion of the analysis model for the Interop project is reported in Figure 4.

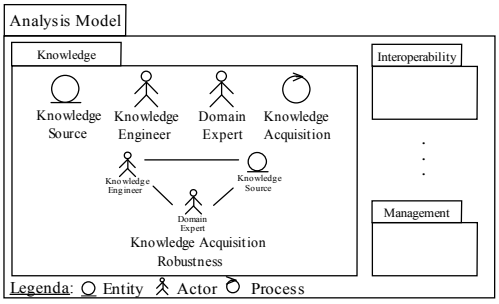

Fig. 4. An excerpt of the analysis model.

Refining the concepts and their relations. At this stage, the gradual and incremental passage from terms to concepts is made clear by the formal definition of relations between sets of synonyms identified in the previous workflow.
As a first structuring step, concepts can be organized in a taxonomic hierarchy through *generalization* (the *kind-of* or *is-a* relation). Three main approaches are known in the literature (Uschold and Gruninger, 1996): *top-down* (from general to particular), *bottom-up* (from particular to general) and *middle-out* (or *combined*). The combined approach consists in finding the salient concepts and then generalizing and specializing them. This approach is considered to be the most effective because concepts "in the middle" tend to be more informative about the domain.
The resulting taxonomy can now be extended with other relations derived from the responsibilities of each concept, as established during conceptual analysis.
The outcome of this step is a UML *class diagram*, using *generalization* (kind-of), *aggregation* (part-of) and *association* relations. A UML association relation can be labelled with a predicate and allows to represent the whole set of relations needed for the ontology being built (see Figure 5 for an example).

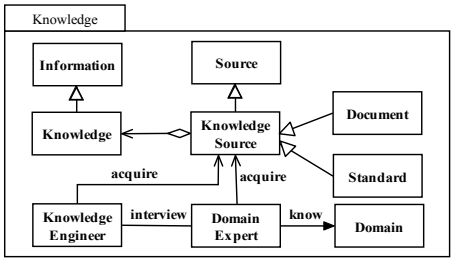

Fig. 5. A portion of the class diagram.

Use-case realization. Use cases identified during the requirements workflow can be realized with the aid of UML *sequence* and *collaboration diagrams*. These diagrams emphasize respectively the sequence and the organization of responsibilities between concepts required for the realization of a use-case. In Figure 6 a sequence diagram is reported.

The outcome of conceptual design is the design model, including class and interaction diagrams (i.e. sequence and collaboration diagrams).

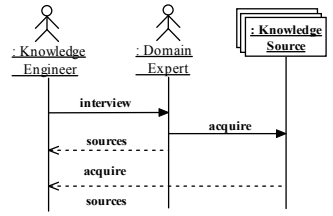

Fig. 6. Sequence diagram of the Knowledge Acquisition use-case (from Figure 2).

3.4 The Implementation Workflow.

The purpose of this workflow is to formalize the ontology in a language and to implement it in terms of replaceable components. Components implement concepts from the design workflow and follow the established grouping into packages (i.e. ontology portions). *Use-case prioritization* from the requirements workflow and *packaging* from all the previous workflows allow component engineers to work on different parts of the ontology to be integrated at subsequent iterations.
Components can be written down in a variety of languages and notations. The adoption of a certain formalism is appropriate as long as it conveys the adequate expressiveness and it allows an easy reuse within the community. As a result of a long standardization effort, the Ontology Web Language (OWL[4]) is the main candidate for encoding an ontology to be used on the Semantic Web.
The outcome of this workflow is the implementation model, including packages of implemented components, each encoding a portion of the ontology.
Figure 7 reports an excerpt of the implementation model for the Interop project.

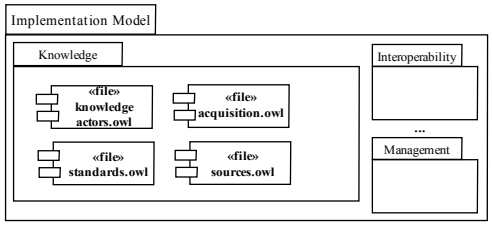

Fig. 7. Part of the implementation model.

3.5 The Test Workflow.

The test workflow allows to verify that the ontology correctly implements its specification. To this end, a number of test cases are developed. A test case is a set of test inputs, execution conditions and expected results for a certain objective, both to verify compliance with a specific requirement and to exercise a particular (knowledge) path through a use case. In table 2 a test case from the Interop project is reported.
In this workflow the modellers assess the compliance of the produced ontology artifact with respect to the user needs captured during requirements. As UPON is a use-case driven process, a test failure means that no use case stresses the required knowledge path (if any) that can satisfy the test. In this case, some requirements integration has to be performed during subsequent iterations.

Table 2 A test case for the Interop project.

Input:	Who acquires knowledge?
Expected Result:	The knowledge engineer alone and through the domain expert.
Conditions:	The domain expert knows the required domain.

4. EVALUATION

In this section we provide a two-fold evaluation of the proposed approach. First, we provide a comparative evaluation with respect to the methodologies introduced in section 2. Second, we briefly describe our experience in using the process in the context of the Interop Project on interoperability and in building an ontology of e-business for the Athena Integrated Project .
In order to evaluate a number of different ontology building processes, Fernández and Gómez-Pérez (2002) present a framework based on the comparison with respect to the IEEE 1074-1995 standard for developing software life cycle processes. Here we integrate UPON into the evaluation framework in order to assess it with respect to the other proposals.
The IEEE standard, applied to ontologies, distinguishes three kinds of processes: *project management processes*, concerning the creation of a project management framework for the entire ontology life cycle; *ontology development-oriented processes*; *integral processes*, required to complete ontology project activities (documentation, evaluation, knowledge acquisition, etc.).
UPON provides full support to the ontology development process, but also to the production of *documentation* (intrinsic to the nature of the process), *evaluation* (through use-case testing), and *knowledge acquisition*.
Because of its nature, UPON does not deal with project management processes and pre/post development activities, while this is a major benefit of the On-To-Knowledge approach (sketched in section 2). On the other side, the adoption of UPON does not require any learning curve for enterprise modellers using UML and the Unified Process, because it is an adaptation of the UP to ontology building. This is an advantage also over the adoption of METHONTOLOGY, that roughly covers the same development processes as UPON. Furthermore, an extension of the UP, the *Enterprise Unified Process* (Nalbone et al., 2004), is being developed with the aim of taking into account project management and

[4] http://www.w3.org/TR/owl-features

all the other pre/post development activities, but this is out of the scope of the paper.

Another big advantage of UPON over the other methodologies is that diagrammation, documentation and versioning can be performed with the aid of a variety of tools specialized for the UP, like Rational Rose, Microsoft Visio, Together ControlCenter, etc.

The methodology is being applied in the context of the Interop European Project to the construction of an ontology of interoperability, aiming at sharing a view of interoperability concepts, integrating the competences from the different areas of Enterprise Modelling, Ontology and Platform & Architectures, and supporting the tasks of manual and automatic classification and retrieval of documents and databases.

UPON is also being applied in the context of the Athena Project for building an ontology of e-procurement, concerning all the processes and the interactions between a buyer and a supplier (e.g., exchange of business documents like an *invoice* or a *purchase order*). The goal of the ontology is to provide a better understanding of the domain of interest and be a support for semantic interoperability between two legacy systems. We envisage three basic uses for it: search and retrieval, reconciliation of exchanged data and processes between business partners.

Although the work is ongoing, a number of clues hint at the success of a methodology based on a well established process coming from the object-oriented software engineering community. The modellers know thoroughly UML diagrammation, the Unified Process and the related productivity tools. Indeed, their familiarity with the process has made it easier to understand and follow UPON during the whole ontology building process.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented UPON, an ontology building methodology that is based on the Unified Process. The guidelines for building an ontology differ from those suggested for developing a software system, but we showed that the same diagrammatic forms can be productively used for each phase of the lifecycle of both software systems and ontologies.

The strength of the approach lies in the UP being a highly customizable framework. It can indeed be tailored to fit a number of variables: the ontology size, the domain of interest, the complexity of the ontology being built, the experience and skill of the project organization and its people. Furthermore, the modellers can decide to adapt the scheme presented here for one of the methodologies derived from the UP (like the Rational Unified Process).

In a future work, we would like to provide a more detailed evaluation of the process with respect to the other proposals as well as an analysis of how to adapt cross-phase activities to the needs of ontology building. In describing UPON, some aspects of the UP, like interfaces, architectures, activity diagrams etc., have been neglected for the sake of space.

ACKNOWLEDGEMENTS

This project is partially funded by the Interop NoE and Athena IP, 6th European Union FP. Special thanks go to Prof. Paolo Bottoni for his stimulating comments on this work.

REFERENCES

T. Berners-Lee, J. Hendler, O. Lassila (2001). The Semantic Web. *Scientific American*, May 2001.

O. Corcho, M. Fernández, A. Gómez-Pérez (2003). Methodologies, tools and languages for building ontologies. Where is the meeting point? *Data & Knowledge Engineering*, **46**, pp. 41-64.

C. Fellbaum, Ed. (1998). *WordNet: an Electronic Lexical Database*, MIT Press.

M. Fernández, A. Gómez-Pérez, N. Juristo. (1997) METHONTOLOGY: From Ontological Art towards Ontological Engineering. *Symposium on Ontological Engineering of AAAI*. Stanford, California.

M. Fernández, and A. Gómez-Pérez (2002). Overview and Analysis of Methodologies for Building Ontologies. *The Knowledge Engineering Review*, **17**(2).

T. R. Gruber (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**, pp. 199-220.

M. Gruninger, and M. S. Fox (1995). Methodology for the Design and Evaluation of Ontologies, Proc. of *Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada.

N. Guarino, M. Carrara, P. Giaretta (1994). Formalizing Ontological Commitments. In *Proceedings of AAAI 94*, volume 1, pp. 560-567.

G. Guizzardi, H. Herre, G. Wagner (2002). Towards Ontological Foundations for UML Conceptual Models. *1st International Conference on Ontologies, Databases and Application of Semantics*, Irvine, California, USA.

I. Jacobson, G. Booch, and J. Rumbaugh (1999). *The Unified Software Development Process*. Addison Wesley, USA.

M. Missikoff, and F. Taglino (2002). Business and Enterprise Management with SymOntoX. 1st Int'l Semantic Web Conference, Sardinia, Italy.

R. Navigli, and P. Velardi (2004). Learning Domain Ontologies from Document Warehouses and Dedicated Websites, *Computational Linguistics* **30**(2), MIT Press, April.

J. Nalbone, M. Vizdos, M. Ambler (2004). *Adopting the Enterprise Unified Process*. White paper, Ronin International Inc.

OntoWeb Deliverable 1.4: A Survey on Methodologies for Developing, Maintaining, Evaluating and Reengineering Ontologies (2002). http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D1.4-v1.0.pdf

Y. Sure, S. Staab, R. Studer (2004). On-To-Knowledge Methodology (OTKM). *Handbook on Ontologies*, Springer, pp. 117-132.

M. Ushold, and M. Gruninger (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, **11**(2).