Ontologies and Software Engineering

Waralak V. Siricharoen

Computer Science Department, School of Science,
University the Thai Chamber of Commerce (UTCC)
Dindeang, Bangkok, Thailand
lak_waralak@yahoo.com, waralak_von@utcc.ac.th

Abstract. This paper is about using ontologies to identify the objects from a problem domain text description. At the center of object models and ontologies are objects within a given problem domain is similar to the concept provided by ontologies. This paper addresses ontologies as a basis of a methodology for object modeling, including available tools, particularly OntoExtract, which can help the conversion process. This paper describes how the developers can implement this methodology on the base of an illustrative example.

Keywords: Ontologies, Software Engineering, Object Models, Artificial Intelligent.

1 Introduction

An ontology, in more generally and well known definition, is a specification of conceptualization [9]. Ontologies described syntactically on the basis of languages such as eXtensible Markup Language (XML), XML Schema, Resource Description Framework (RDF), and RDF Schema (RDFS). The object oriented paradigm is the framework in software engineering, influencing all effort in information science. Discovering the right objects seems to be the most difficult task in the whole development process. Object oriented software development is well supported by numbers of working methods, techniques, and tools, except for this starting point - object identification and building the related system object model. Converting the text description of system problem domain and respective functional requirement specifications into an object model is usually left to the intuition and experience of developers (system analysts). Recently there has been great research interest in applying ontologies for solving "language ambiguity problem" as either an ontology-driven or ontology-based approach [9]. This is true for object oriented software engineering, mainly because of the similarity in the principles of the two paradigms. Moreover, the object systems similar to ontologies, which represent conceptualized analysis of a given domain, can be easily reused for different applications [10]. An ontology is a specification of a representational vocabulary for a shared domain of discourse: definitions of classes, relations, functions, and other objects [1] or, more generally, a specification of conceptualization [2]. Semantic web uses ontologies as a tool for easy integration and usage of content by building a semi-structured data

model. To solve the problem of heterogeneity in developing software applications, there is a need for specific descriptions of all kinds of concepts, for example, classes (general things), and the relationships that can exist among them, and their properties (or attributes).

The proposed methodology described in this paper, is based on five submodels, only two namely the text description model (T-model) and class (object) model (C-model), are included in the classical object oriented software development process. The others model used represent specific analysis work, which the developers should do, to get benefit from using ontologies for identification of objects. The basic idea is to ensure suitable transformation of the models from one to another using respective procedure and tools. This paper is structured as follows: section 2 is dedicated to a more detailed description of the models as well as to discussion on the techniques and tools, which can be practically used for model transformation. An illustrative example of a part of the information system for the domain of academic management is used throughout the paper to support the explanations; finally, section 3 summarizes the proposed methodology.

2 The Proposed Methodology

The proposed methodology is based on integration of five submodels (T-model, O-model, OL-model, I-model, and C-model). Models are inseparable and one of the most significant parts of any methodology. They help developers to better understand complex tasks. Object oriented analysis of a system under development is a good example of such a complex task. The complexity stems from the fact that in object oriented development everything is based on objects but their identification in a given problem domain is completely left to the intuition of the developer. Fig. 1 shows the basic idea of the models used and transformation process on them. The details of models are described as follows.

2.1 T-Model: Text Description Model

The starting point of the objects identification is the T-model, which represents a concise description of the problem domain, where the software system under development will work, written in English. If not available the T-model is a deliverable from a system analyst's work on the general user requirements for the system functionality. The presumption is that this problem domain description contains the main objects. This text is represented as an ontology description after processing by an ontological engine tool in our case *Corporum OntoExtract* [3]. It is a web-based version of *Corporum*, which is able to extract ontologies and represent them in XML/RDF/OIL (default in RDF schema) and also to communicate with and negotiate the final format of the to-be-submitted ontology extracted from a specific text [4]. This tool can interpret text, in the sense that it builds ontologies that reflect world concepts as the user of the system sees and expresses them. So at this point in the process, the text is automatically processed and converted into ontologies, which can be done online. To help this process we refer to a tool of conceptualization - an

ontological engine, which applied on the T-model, generates an ontological description (O-model) of the problem domain at hand.

2.2 O-Model: Ontological Model

The ontology described in RDFS defines the names and relations of the extracted concepts, or object names. RDFS provides a mechanism to define domain-specific properties and classes of resources to which developers may apply those properties [5]. More specifically, an ontology description is recognizable as an ontology language. Classes are specified with `<rdfs:class>`. Subclasses and subproperties are specified using `<rdfs:subClassOf>` and `<rdfs:subPropertyOf>` (the top class defined in the schema is “Resource”) respectively. When a class is a subclass of several superclasses, this is interpreted as a conjunction of superclasses [7]. A class may also be defined as a subclass of other classes if evidence is found that the class is indeed a subclass. A subclass relationship found by this tool is based on information about the term [3].

Fig. 1. The proposed methodology of objects identification using ontologies

An important category that is exported by the CORPORAUM OntoExtract engine is the *cross-taxonomic* relations. While a typical ontology often represents taxonomy,

<isRelated> refers to cross-taxonomic links that may exist within a domain and, if represented, can make a difference in finding needed information based on context descriptions. In short, it can identify the possible relations between objects.

2.3 OL-Model: Ontologies Library Model

Data (attributes) and functions (methods, operations) are the two fundamental parts of any objects. We can continue in this way relying on the decision making abilities of the developers to the final acceptable object model of the system. However, because of the requirement for decision making this process can still be characterized as subjective or even intuitive, which was the main reason to propose these models. To avoid this situation we can recall the most powerful feature of both object and ontology orientation - they allow for a high degree of reusability of their artifacts in different application domains. The idea is very simple - if something is defined already and checked successfully and has been used in practice, perhaps with some adjustments it can be used for another developer's needs. This idea is implemented and used broadly in object oriented software engineering through business objects and related patterns, shown in more detail for example in Batanov and Arch-int, 2003. This paper proposes here an extension of this idea introducing the notion of Ontological Business Object Pattern (OBOP). An OBOP is an ontology-based description of a business object that presumably will be included as a working object in the object oriented software system. It actually rely on the fact that there are a great number of ontological descriptions of concepts (objects) in different problem domains, existing already and available from ontology library systems such as WebOnto, Ontolingua, DARPA Agent Markup Language (DAML), Simple HTML Ontology Extensions (SHOE), etc.

The DAML ontology library and SHOEntity library, is more specifically their catalogs of ontologies, which are available in XML, HTML, and DAML formats. Here classes are called categories and these categories constitute a simple "is-a" hierarchy while the slots are binary relations [8]. What the developer should do at this phase is to select the suitable ontology for the respective problem domain. Fig. 2. shows an example of how available ontological description for our particular problem domain can be considered as OBOP. Representation of ontology specifications is standardized in a form of object description and this provides a great advantage for software developers. For example, the ontological description shown in Fig. 2. is found in the ontology library and has a structure, which can be used by the developer directly as not only class hierarchy but as a structured content of respective classes. Therefore, this description can be considered as OBOP. Within this pattern the concept (object) "student" possesses exactly the properties (attributes) necessary for the system under development. We can say the same for the root concept (object, class) "person". Moreover, in the ontology the attributes themselves are treated as concepts (objects) just like in object orientation, which means that we can follow and extract the description of all objects that we are interested in within the class hierarchy.

Fig. 2. Example of ontological class hierarchy used as a pattern of computer science department ontology version 1.1 [6]

2.4 I-Model: Integrated Model

From O-model, we capture some objects from text description, on the other hand, OL-model also provide some relevant existing concept within ontologies library. I-model play the important role to mapping the objects appear in only both models to be working object in this particular problem domain. In order to emphasize the necessity of this model we will review what information the developer has up to this point working with the models described above:

1. Set of objects in the problem domain $PD = \{O_1, O_2, O_3, \dots, O_a\}$ with their names and relationships, extracted from the T-model by an ontological engine (OntoExtract).
2. Set of objects $FOE = \{O_1, O_2, O_3, \dots, O_b\}$ with their names and relationships as a result of applying an ontological engine.
3. Set of objects $BOP = \{O_1, O_2, O_3, \dots, O_d\}$ with their names, relationships (including hierarchical information) and functions as a result of searching for OBOPs in ontology libraries (DAML and SHOEntity).

Fig. 3. shows in graphical form, although far from precise, the existing situation. Without a doubt all objects are within the system problem domain but on one hand their number is still large and they are defined from different perspectives. The presumption, based on a number of experiments, is that the basic objects, which will play a substantial role in ensuring the system functionality, will appear in the above models regardless of the perspective. This practically means that we can apply a simple integration procedure - intersection of the above sets - to identify those objects In Fig. 3. the resulting area is O, or

Fig. 3. Integration Model

$$\mathbf{O} = \mathbf{PD} \cap \mathbf{FOE} \cap \mathbf{BOP}$$

The final decision should be taken by the developer.

2.5 C-Model: Class Model

In the O area in Fig.3 from I-model consist of the interesting objects which will appear in C-model. The C-model is the goal of preliminary analysis of object oriented systems. This is the well-known class hierarchy representation, including some initial but significant relationships for the system functionality contents of objects – data and behavior (functions, operations). We stress on the word initial here to emphasize the fact that the analysis is far from over yet. The developer should continue applying the conventional analysis models, methods and techniques on the C-model, which can lead to substantial changes, including adding new objects, deleting some objects, adding or removing some elements of the included objects, etc. The C-model can be represented graphically using different tools such as Rational Rose (class diagrams), textually using either some natural language or pseudo programming language, and finally using some highly structured tag-based language.

3 Conclusions

The ultimate goal of the developer's efforts is actually the objects identification which is C-model. This is because the objects included the C-model should contain the possible objects necessary for the next phases of design and implementation of the software system. It is clear now the already mentioned problem with "language ambiguity" - different interpretations of the T-model, without any formal support of the choice of participating objects, would lead to creating C-models, which are quite probably inconsistent, incomplete or inefficient for the further steps of design and implementation. We believe that using ontology as a tool of conceptualization working on the T-model can make (if not fully formal at least) semi-formal the

process of creating the C-model and in this way help developers in this complex and imprecise task. This is the major motivation of the work described briefly in this paper.

The author believe that merging ontologies with existing methods, techniques, and tools used during the analysis phase of complex object oriented software systems can contribute significantly to reaching better decisions, with a positive effect on all the subsequent phases of the development process. This paper describes a methodology for supporting the analysis phase of object oriented software engineering using ontologies for identification of system objects. Five submodels are introduced and briefly described in the paper as a part of this methodology. The author also believe that these models and the process of their transformation can help developers of complex object oriented software systems to: (a) transform user requirements (represented as text description) into an object model of the system under development based on the use of ontologies; (b) improve the effectiveness and efficiency of the existing methodology for high-level system analysis in object oriented software engineering.

References

1. Gruber, T.R.: Towards Principles for the Design of Ontologies Use for Knowledge Sharing, In Proceedings of IJHCS-1994, Volume 5 No.6 (1994) 907-928
2. Cullot, N., Parent, C., Spaccapietra, S., and Vangenot, C.: Ontologies : A contribution to the DL/DB Debate. Available online: downloaded on January (2005)
3. Engles, R.: Del 6: CORPORUM – OntoExtract ontology extraction tool, On-To-Knowledge: Content-driven knowledge management tools through evolving ontologies. In IST project IST-1999-1032, On-To-Knowledge (2001)
4. Engles, R.H.P., Bremdal, B. A., and Jones, R.: CORPORUM: a workbench for the semantic web, In EXML/PKDD workshop, CognIT a.s., (2001)
5. Klein, M.: XML, RDF and relatives, In IEEE Intelligent Systems March/April (2001) 26-28
6. Heflin, J.:SHOE Ontologies, Available online: <http://www.cs.umd.edu/projects/plus/SHOE/onts/cs1.1.html>, downloaded on January (2007)
7. Gil, Y., and Ratnakar, V.: A comparison of (semantic) markup languages, In Proceedings of the 15th International FLAIRS Conference, Special Track on Semantic Web, Pensacola, FL (2002)
8. Noy, N., Sintek, F. M., Decker, S., et al.: Creating semantic web contents with Protégé-2000, In IEEE Intelligent Systems, March/April (2001) 60-61
9. Deridder, D., and Wouters, B.: The Use of Ontologies as a Backbone for Software Engineering Tools, Programming Technology Lab, Vrije Universiteit Brussel, Brussels, Belgium (1999)
10. Swartout, W.: Ontologies”, In IEEE Intelligent Systems, January/February (1999) 18-25