Use of Ontologies in Software Engineering

Emdad Ahmed
Integration Informatics Laboratory
Department of Computer Science
Wayne State University
Detroit, Michigan, USA
emdad@wayne.edu

Abstract

This paper surveys the role ontologies might play in the whole software development life cycle. Use of ontologies in each phase of software development yields competitive advantage over traditional approach enabling greater information sharing and reuse. In this survey paper, we present a concise description of various ontology-based approaches in Software Engineering ordered by their position in the Software Engineering life cycle. We also report ontology editor tool Protege as part of our survey.

Keywords: ontology, conceptual modeling, software engineering, knowledge engineering.

1 Introduction

Software quality and productivity can be improved by the use of Software Engineering Environment (SEE)s. Domain Oriented Software Engineering Environments (DOSEEs) are a special class of SEE{s} that uses domain knowledge to guide software developers across the several phases of the software process. DOSEEs organize the application domain knowledge facilitating the understanding of the problem during system development [22]. Ontologies are a good way to describe and organize domain knowledge in a DOSEE. However, building ontologies involves the specification of concepts and relations that exist in the domain, besides their definitions, properties and constraints. Software development is a complex task and thus it is essential to provide tool support for it. Stand alone CASE tools were the first initiative to provide this kind of support. Although these tools had significantly affected the practice of software development, their potential was limited by the difficulties involving in integrating them. This fact gave rise to the research SEE{s}, which are integrated collections of tools that facilitate software engineering, supporting its activities across the software life cycle [15]. SEE{s} have a history of about three decades, starting from supporting small fragments of the software process, until achieve the notion of process-centered SEE{s} [11]. Throughout this history, integration is the main challenges, seems to be more and more complex [23, 13]. In several fields of Informatics such as AI, IS, DB or Web technology, the term "ontology" has widely been used during the last years. Various languages and techniques for the definition and use have been published and disseminated. Recently, the ontology approach is being considered in parts of the SE mainly for its proximity to the conceptual modeling fields, to the UML language and the fields of software component re-use and integration.

One of the objective of this survey paper is to demonstrate why ontologies are important and can be very useful for conceptual modeling. In this survey paper, we present a concise description of various ontology-based approaches in SE ordered by their position in the SE life cycle. We also surveyed Protege, an ontology editor developed in the context of ODE(Ontology-based Software Development Engineering) and discuss how ontologies are being used in ODE. We have been working on developing a prototype web data integration tool which will run ad hoc user query. Ontologies are the core of the system. Details are in [2, 3]. We also materialize the web response in a relational database [1]. [9] discussed a number of purposes for which ontologies are being used. Out of those purposes, our target is to provide a mediating mechanism for accessing heterogeneous data and information sources, particularly on the Web. Due to space limitation, we omit the details of those work.

The rest of the paper is as follows: in section 2 we discuss background of ontologies, software engineering, knowledge engineering, some related work. Section 3 deals with the role of ontologies in the software engineering lifecycle. Section 4 categorize ontologies in software engineering, section 5 deals with some advantages of ontologies in SE, The paper ends with conclusions in section 6.
2 Background

2.1 Ontologies

According to Gruber’s definition an ontology is "a formal explicit specification of a shared conceptualization". By "explicit" we mean that the concepts used and the restrictions applied to them are clearly defined. Later authors have considered it important to add to this definition two new requirements: that the said specification be (1) formalized and (2) shared. By "formalized" it is meant that a machine can process it. By "shared" it is understood that the knowledge acquired is the consensus of a community of experts. An ontology mainly captures three types of relationship: 

- synonym
- isa
- partof

According to the generality level, the following ontology types exist [24]:

- High level ontologies
- Domain ontologies
- Task ontologies
- Application ontologies

The following alternative classification is also found in the literature:

- Generic or common-sense ontologies
- Representational ontologies
- Domain ontologies
- Method and task ontologies

Figure 1: Kinds of ontologies according to the generality level

1 [24] shows ontology classification. Of the utilities of ontologies in any field of human activity, the followings are the principal uses of ontologies in Software Engineering:

- Clarify the knowledge structure
- Reduce conceptual and terminological ambiguity
- Allow the sharing of knowledge

2.2 Software Engineering

Software Engineering is the "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software". Software Engineering has undergone fundamental changes during the last three decades. In order to cope with the complexity inherent to software, there has been a constant drive to raise the level of abstraction through modeling and higher-level programming languages. However, many problems have only partially been solved including component reuse, composition, validation, information and application integration, software testing and quality. Such fundamental issues are the motivation for new approaches affecting every single aspect in Software Engineering.

2.3 Knowledge Engineering

The engineering of knowledge-based systems is a discipline which is closely related with Software Engineering. The term Knowledge Engineering is often associated with the development of expert-systems, involving methodologies as well as knowledge representation techniques, e.g frames, slots etc. Since its early days the notion of "ontology" in Computer Science has emerged from that discipline, giving rise to Ontology Engineering, which is the focus of this survey paper. Due to the emergence of the "semantic web" vision, ontologies have been attracting much more attention recently.

2.4 Ontology vs Conceptual Model

In the SE and IS communities, perhaps due to the historical importance of conceptual modeling, there is frequent confusion between ontology and conceptual models. In some sense, an ontology has a similar function to a database schema because the first provides meta-information that describes the semantics of the terms or data, but there are several important differences between these concepts [24]:

- Languages for defining and representing ontologies (OWL, etc.) are syntactically and semantically richer than common approaches for databases (SQL, etc.).
- The knowledge that is described by an ontology consists of semi structured information (that is, texts in natural language) as opposed to the very structured data of the database (tables, classes of objects, etc.).
- An ontology must be a shared and consensual conceptualization because it is used for information sharing and exchange. Identifiers in a database schema are used specifically for a concrete system and do not have the need to make an effort to reach the equivalent of ontological agreements.
- An ontology provides a domain theory and not the structure of a data container.

Again, [26] discussed how ontology is different from object-oriented modeling as well.
2.5 Related Work

2.5.1 Ontologies to represent real world knowledge

It is believed that the use of ontologies will fundamentally change the way in which software systems are constructed and that software designers will have libraries of ontologies from which they can choose appropriate ones [30]. The use of ontologies is found in many areas such as CYC project [20] and in natural language understanding. Work in database design automation has focused on developing ontologies for classifying entities and relationships [28]. Most ontology creation is carried out on a manual basis. Most of the current tools support only certain aspects of ontology creation and manipulation and are application specific. For example the RiboWeb [4] system incorporates ontology constructs related to storing molecular information.

Few works have explored semantic issues in SEEs. Brown and McDermid [6] were one of the firsts to talk about that. They proposed a classification of tool integration levels that includes what they call "semantic level" which could be achieved through including metadata in the SEE’s repository. Oliveira et al.[23] proposed the use of ontologies in Domain Oriented SEEs. Sugumaran et al. [29] proposes a methodology for creating and managing domain ontologies. The methodology is heuristic-based and focuses on identifying and defining the terms, properties, relationships and constraints needed to model an application.

2.5.2 Ontology frameworks

A number of frameworks for creating ontologies have been proposed with the most well known being SHOE [16] and Ontolingua which is an ontology representation scheme from Stanford University that uses a Lisp-like notation to express ontologies.

2.5.3 Ontology Editor

There are many ontology editors described in the literature. OntoEdit [27] pursues the modeling of ontologies such that graphical means exploited for modeling of concepts and relations scale up to axiom specifications using RDFS. OILEd [25] supports the construction of ontologies in Ontology Inference Layer (OIL). The Java Ontology Editor (JOE) [19] was developed to help users build and browse ontologies. Musen et al. [21, 12] discussed the use of software engineering methodology Protege, specially in the area of medical informatics. Protege-2000 aims to support knowledge acquisition and to reach interoperability with other knowledge representation systems. It has classes, instances of these classes, slots representing attributes of classes and instances, and facets expressing additional information about slots. Protege-2000 generates knowledge-acquisition forms automatically based on the types of the slots and restrictions on their values allowing ontology instantiation.

2.5.4 Integrating ontologies into CASE tools

In [10] an overview is given of a number of desirable CASE tool properties. The most important amongst them is that a CASE tool should provide a central repository. Enhancing the central repository with an ontological engine will be particularly useful. [8] integrated an ontological engine into the Rational Rose CASE tool. The data used by the ontological engine was WordNet 1.6. Almeida Falbo et al. [7] discussed how ontologies are used in ODE, an Ontology-based software Development Environment to make it a Semantic Software Engineering Environment (SSEE).

3 Ontologies in the Software Engineering Lifecycle

3.1 Analysis and Design

During this phase of Software Engineering life cycle, two main areas of applications have been identified in which ontology can play a vital role. First, requirement engineers can benefit from ontologies in terms of knowledge representation and process support. Second, component reuse is chosen as a potential application area during design.

3.1.1 Requirements Engineering

An ontology can be used for both, to describe requirements specification documents and formally represent requirements knowledge [5]. In most cases, natural language is used to describe requirements, i.e in the form of use cases. However, it is possible to use normative language or formal specification languages (such as Z, B [31] etc.) In contrast to traditional knowledge-based approaches, e.g formal specification languages, ontologies seem to be well suited for an evolutionary approach to the specification of requirements and domain knowledge. Moreover, ontologies can be used to support requirements management and traceability, which help software engineers understand the relations and dependencies among various software artifacts.
3.1.2 Component reuse

Usually the search for reusable components takes place after the analysis phase, when the functional requirements are settled [17]. Since most reuse repositories are limited to a plain syntactical keyword based search, they suffer from low precision (due to homonyms) and low recall (due to synonyms). Ontologies can help in this regard to describe the functionality of components using a knowledge representation formalism that allows more convenient and powerful querying [18].

3.1.3 Implementation

A critical step in the software development process is moving from analysis and design to implementation. The question arises how ontologies can be leveraged to narrow the gap between design and implementation. Two areas of interest are the overlaps of software modeling with ontology languages and the run time usage of ontologies in applications. Integrating with Software Modeling Languages There are several alternatives for integrating MDA-based information representation languages and ontology languages. Whereas some regard the UML as ontology representation language, others employ the UML as modeling syntax for ontology development. UML-based tools can be extended more easily to support the creation of domain vocabularies and ontologies. Ontology as Domain Model Since a domain model is initially unknown and changes over time, a single abstraction and separation of concerns is considered feasible if not necessary. Therefore a single representation of the domain model should be shared by all participants throughout the life cycle to increase quality and reduce costs. The mapping of a domain model to code should be automated to enable the dynamic use by other components and applications.

3.1.4 Coding Support

It is the practice in OO programming to decouple interface specification (like Java Interface, abstract class) from its implementation in order to make requesting applications independent from internal modifications. Nowadays the programmers have to deal with a large number of APIs, thus the documentation of those APIs have become an important issue.

3.1.5 Code Documentation

To help organize generated artifacts and maintain a historical overview of the process, a ontology-based tool is needed that uses ontologies to combine real world domain information and software engineering knowledge, to produce up-todate documentation that evolves over time. Now-a-days what is really used by software maintainers are the UNIX/LINUX style grep search facilities. Linking between documentation and source code ontology is highly desirable.

3.2 Deployment and Run-time

In modern three-tier architectures (e.g in Web application front end client GUI, web server and backend database server), the middleware layer lies in the focus of attention. Issues like interdependencies between modules or legal constraints make the management of middleware systems a cumbersome task. In this case ontologies can provide a mechanism to capture knowledge about the problem domain. In most software systems, the "business logic" are hard coded in programming languages. Thus changes to the business logic of a software system require modifications to the source code. Since many companies are facing flexible, frequently changing business environments nowadays, technologies are sought that can support a quick propagation of new business rules into the core software systems. Business rule engines are a possible solution to this problem. The basic idea is to separate business logic and processing logic. A very good example in this context is the J2EE deployment descriptor like web.xml file.

3.3 Maintenance

3.3.1 Project Support

Dhruv [14] is a semantic-web enabled prototype to support problem-solving processes in web communities. The application scenario is how open source communities deal with bugs e.g Bugzilla in their software under development. Ontologies help to connect the electronic communication of the developers with bug-reports and the affected areas in the source code. Central concepts are the community, their interactions and content.

3.3.2 Testing

Software is not complete unless documented and provided with test data and a test mechanism. Ontologies can help to generate basic test cases since they encode domain knowledge in a machine processable format. A simple example for this would be regarding cardinality constraints. Since those constraints define restrictions on the association of certain classes, they can be used to derive equivalency classes for testing.
Categorizing Ontologies in Software Engineering

[14] proposed two dimensions of comparison to achieve a more precise classification. Putting these two dimensions together, the matrix shown in the following figure results. We see four basic areas there: **Ontology-driven development (ODD)** subsumes the usage of ontologies at development time that describe the problem domain itself. Prime example are the approaches in the context of MDD. **Ontology-enabled development (OED)** also uses ontologies at development time, but for supporting developers with their tasks. For example, component search or problem-solving support can be put in here. **Ontology-based architectures (OBA)** use an ontology as a primary artifact at run time. The ontology makes up a central part of the application logic. Business rule approaches are an example for this kind of application. **Ontology-enabled architectures (OEA)** finally, leverage ontologies to provide infrastructure support at the run-time of a software system. An example is semantic web services composition.

![Figure 2: Usage categories for ontologies in Software Engineering](image)

Advantages of Ontologies in Software Engineering

Since modeling ontologies is a tedious and costly task, it is always important to demonstrate the advantages one can gain by applying ontologies in SE. This is underlined by the fact that most of the formal foundations of ontologies have been in place for a long time without enjoying a wide spread adoption by software engineers. The advantage of an ontology is that we are dealing with concepts and getting rid of several Software Engineering problems usually linked to natural language vocabularies like synonymy, homonymy, polysemy, etc. The main properties of an ontology are sharing and filtering. An ontology defines what should be extracted from a system in order to build a given model of the system. Whenever a software development project is not just concerned with a specific application but is part of a landscape of projects located in the same domain, an ontology based approach might be worth considering and even be advantageous to a traditional approach.

Another important factor is the flexibility of ontologies. With information integration as a major use case, ontologies are well suited to combine information from various sources. Also the flexibility allows to extend existing ontologies very easy, thus fostering the reuse of existing work. This is further promoted by the web focus of current ontology approaches. Due to the fact that software systems also get increasingly web enabled and must thus cope with data from heterogeneous sources that may not be known at development time, software engineers seek technologies that can help in this situation. Regarding more SE specific advantages, ontologies make domain models first order citizens. While domain models are clearly driving the core of every software system, their importance in current SE processes decreases after the analysis phase. The core purpose of ontologies is by definition the formal descriptions of a domain and thus encourages a broader usage throughout the whole SE lifecycle.

Conclusion

There are a lot of interest and discussion about how ontologies and SE fit together and how both the communities can learn from each other. In this paper we presented a couple of case studies that use ontologies in a SE context. We selected examples from the entire SE life cycle. It is observed that the purpose of ontologies as well as the real benefits are hard to grasp without a proper framework for analysis. It is possible to move on from basic ontology research of representation and formalization issues towards research focused on advanced applications of ontologies. The longer the ontology is used, the more domain knowledge is present in the ontology and less effort will have to be spend in adding data to the ontology. Consequently development of applications will take a lot less time and debugging effort as more knowledge is reused.
Acknowledgements

The work is partially supported by Wayne State University, Computer Science Department conference, travel fund.

References


