©2010 International Journal of Computer Applications (0975 – 8887)
Volume 1 – No. 20

Ontology Driven Software Engineering in Multi-Site Software Development

Prof. Rashmi Phalnikar
Asst. Prof., IT Department
MIT COE Pune

Prof. S.D. Joshi
Prof., Computer Department
BV COE Pune

ABSTRACT
Globalization of software development enables multiple teams residing in cities and countries to work together in a networked distributed fashion. However, the diversity between the software teams, their members, team leaders and managers can give rise to several practical problems and unidentified issues. The diversity arises due to lack of face to face communication, different cultural and educational background and varied interpretation methods.

Ontology is an explicit specification of a conceptualization. Ontology drive software engineering offers a direction towards solving the inter-operability problems brought about by semantic obstacles, i.e. the obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontology for a particular domain.

This paper attempts to understand Ontology, how it can benefit multi site

Index Terms:
Ontology, Software Engineering, Multi site development

Keywords
Keywords are your own designated keywords which can be used for easy location of the manuscript using any search engines.

1. INTRODUCTION:
The definition of Ontology was originally proposed in 1992 by Gruber. The body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them (Genesereth & Nilsson, 1987). A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly. [1] The term has its root in philosophy, where Ontology is a systematic account of existence.

Ontology driven Software engineering refers to the different ways in which ontologies (i.e., formalized conceptual models of real world domains) contribute to improving Software Engineering – its processes and its artifacts. It encompasses following

(1) Ontological principles can be used as the basis of improved development languages;

(2) Ontologies can help improve the way in which software development projects are organized

(3) Ontological domain models can drive or refine typical development phases, such as requirements, design and implementation.

2. Need for Ontology driven Software Engineering
The vision is informally and incompletely specified and subject to frequent changes that leads to the Business-IT gap. This gap makes it difficult for IT to give a reasonable estimate of time and cost for the project and impacts greatly on the business case.

The inconsistency in presentation, documentation, design and diagrams could prevent access by other teams or different approaches to communication and implementation. own individual guide, and when they communicate, their own knowledge base and terminology is different from those of others. Other issues like diagrams with no standard notation can create chaos in the development life cycle. [1] Therefore, in the field of Software engineering, Ontology is used to refer to what exists in a system model. An ontology, in the area of computer science, represents the effort to formulate an exhaustive and rigorous conceptual schema within a given domain, typically a hierarchical data structure containing all the relevant elements and their relationships and rules (regulations) within the domain.

Ontology-Driven Software Engineering refers to the different ways in which ontologies (i.e., formalized conceptual models of real world domains) can contribute to improving Software Engineering - its processes and its artefacts.

Ontologies have the potential of significantly impacting diverse aspects of software development. For example:

(1) ontological principles can be used as the basis of improved development languages;

(2) ontologies can help improve the way in which software development projects are organized;

(3) ontological domain models can drive or refine typical development phases, such as requirements, design and implementation.

Important work is already been carried out by researchers and practitioners of leading organizations (e.g., by working groups at the OMG and the W3C).

E.g., consider a banking ontology with a rule that identifies a customer by its (unique) Customer ID. All applications that commit to this interpretation of this ontology need to satisfy the identification rule. Any bank applications that do not foresee a Customer ID for every customer will not be able to commit or reuse the banking Ontology. Without such a banking ontology, two

applications would even not be able to communicate (no sharing of vocabulary and domain rules by two applications**).**

3. Comparison of Data Modelling, UML and Ontology Engineering

Unlike data models, the fundamental asset of ontologies is their relative independence of particular applications, i.e. ontology consists of relatively generic knowledge that can be reused by different kinds of applications/tasks. Ontology contains the vocabulary (terms or labels) and the definition of the concepts and their relationships for a given domain. Domain rules restrict the semantics of concepts and conceptual relationships in a specific conceptualization of a particular application domain. These rules must be satisfied by all applications that want to use – or "commit to" an interpretation of – a domain.

A data model, on the contrary, represents the structure and integrity of the data elements of the, in principle "single", specific enterprise application(s) by which it will be used. Therefore, the conceptualization and the vocabulary of a data model are not intended a priori to be shared by other applications [6]

Data modelling in software engineering is the process of creating a data model by applying formal data model descriptions using data modelling techniques. Data modelling is a method used to define and analyze data requirements needed to support the business processes of an organization.

The data requirements are recorded as a conceptual data model with associated data definitions. Actual implementation of the conceptual model is called a logical data model.

UML: Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of software engineering. UML includes a set of graphical notation techniques to create visual models of software-intensive systems.UML combines best techniques from data modelling (entity relationship diagrams), business modelling (work flows), object modelling, and component modelling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

Conceptual (or Ontology) modelling deals with the question on how to describe in a declarative and abstract way the domain information of an application, its relevant vocabulary, and how to constrain the use of the data.

The domain knowledge is separate from instance knowledge. The instance knowledge varies depending on its use for a particular project. [5]

4. Web Ontology Language OWL:

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies, and is endorsed by the World Wide Web Consortium.OWL ontologies are most commonly serialized using RDF/XML syntax. OWL is considered evolution of the project by proposing concrete changes to the ontology.[5] One of the main advantages of OWL is that it is a declarative language with a formal syntax and semantics. As such it can be used unambiguously by computer programs. one of the fundamental technologies underpinning the Semantic Web, and has attracted both academic and commercial interest.

The Web Ontology Language, OWL, was developed to facilitate greater machine interpretability of human knowledge by providing additional vocabulary along with formal semantics.[5]

There are several ontology languages available such as Resource Description Framework (RDF) , Web Ontology Language (OWL), DARPA Agent Markup Language (DAML), Ontology Interchange Language (OIL), DAML+OIL, Simple HTML Ontology Extensions (SHOE) etc. for capturing knowledge of interest. Different ontology languages have different facilities. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C) (http://www.w3.org/). It has the most complete set of expressions for capturing the different concepts and relationships that occur within ontologies; therefore, the software engineering knowledge is captured in OWL.

A typical application may uses ontology as given in Figure 1. It may be a wasted effort to express specification and then to directly add them to the application. OWL will help to express this declarative language in a formal syntax and semantic. And hence result in less ambiguity.

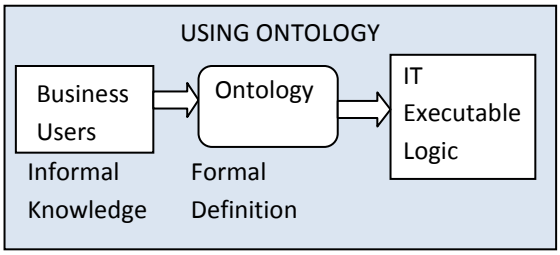

Fig. 1: Use Of Ontology

i.OWL Ontologies:

The data described by OWL ontology is interpreted as a set of "individuals" and a set of "property assertions" which relate these individuals to each other. OWL ontology consists of a set of axioms which place constraints on sets of individuals (called "classes") and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. For example, an ontology describing families might include axioms stating that a "hasMother" property is only present between two individuals when "hasParent" is also present, and individuals of class "HasTypeOBlood" are never related via "hasParent" to members of the "HasTypeABBlood" class. If it is stated that the individual Harriet is related via "hasMother" to the individual Sue, and that Harriet is a member of the "HasTypeOBlood" class, then it can be inferred that Sue is not a member of "HasTypeABBlood".

The Web Ontology Language, OWL [5], was developed to facilitate greater machine interpretability of human knowledge by providing additional vocabulary along with formal semantics. The language forms a knowledge continuum between Business and IT, and provides a mechanism by which the Business can drive the evolution of the project by proposing concrete changes to the ontology.[5] One of the main advantages of OWL is that it is a declarative language with a formal syntax and semantics. As such it can be used unambiguously by computer programs.

ii. Rules Of Ontology

For knowledge engineering methodology for developing ontology, there are some fundamental rules in ontology design. These rules may seem rather dogmatic. However, these rules can often help in making design decisions:

- There is no one correct way to model a domain – there are always viable alternatives.
- The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

A practical example is described in the paper "Ontology Driven Software Engineering for Real Life Applications" authored by Michel Vanden Bossche1, Peter Ross2, Ian MacLarty2, Bert Van Nuffelen1, and Nikolay Pelov1 [5].They have used ODASE to build a 250 person month e-insurance project for a multi-national insurance firm, where only 35% of the requirements were known at kickoff. They required one third of the time of the next closest quote for the project, and a similar project built classically at another insurance firm required also around three times the resources.

Providing more ontology rules, which are important for effective and meaningful interoperation between applications, may limit the genericity of ontology. However, light ontologies, i.e. holding none or few domain rules, are not very effective for communication between autonomous software agents. Fig. 2 explains the Software Engineering Ontology :

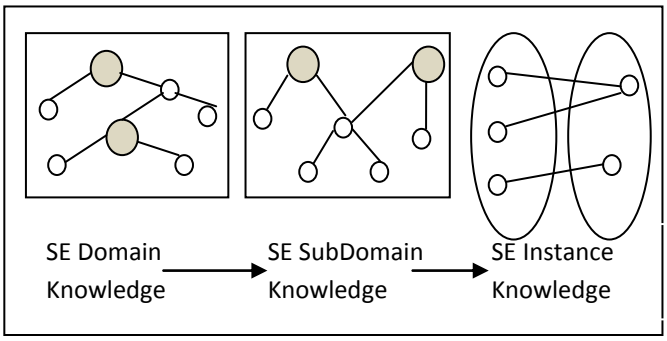

Fig. 2 Software Engineering Ontology

Example of transformation using ontology:

Software engineering ontology instantiations are derived as a result of populating software engineering project information and are referred to as ontology instances of software engineering ontology classes. The transformation process is usually accomplished by mapping various project data and project agreement to the concepts defined in the software engineering ontology.

5. Applications :

Ontology helps in developing a shared understanding across a project. Ontology represents a consensual, shared description of the pertinent objects considered as existing in a certain area of knowledge. They constitute a special kind of software artifact conveying a certain conception of the world. This is specifically designed with the purpose of explicitly expressing the intended meaning of a set of agreed existing objects. The software engineering ontology defines common sharable software engineering knowledge including particular project information. Software engineering ontology typically provides software engineering concepts – what they are, how they are related, and can be related to one another – for representing and communicating over software engineering knowledge and project information.

6. Benefits:

Ontology in Software Engineering will:

- Provide a source of precisely defined terms that can be communicated across people, organizations and applications (information systems or intelligent agents)
- Offer a consensual shared understanding concerning the domain of discourse
- States explicitly all hidden assumptions concerning the objects pertaining to a certain domain of knowledge.
- Early feedback to determine the trade-off between delay and functionality, a key requirement when time-to market is critical.
- Possible to exploit the same information in different ways.
- Saving on the almost 80% of an IT budget that is spent doing corrective and adaptive maintenance. As compared to the increased flexibility which was required for being able to react quickly to new requests of the Business during the development
- Identify weaknesses in the model early and consequently propose new concepts to capture the intended business meaning.

These concepts facilitate common understanding of software engineering project information to all the distributed d members of a development team in a multi-site development environment. Reaching a consensus of understanding is of benefit benefit in a distributed multi-site software development environment. Software engineering knowledge is represented in the software engineering ontology whose instantiations are undergoing evolution. Software engineering ontology instantiations signify project information which is shared and has evolved to reflect project development, changes in software requirements or in the design process, to incorporate additional functionality to systems or to allow incremental improvement**.**

6. CONCLUSION

In conclusion, the software engineering ontology facilitates collaboration of remote teams in multi-site distributed software development. We have explored software engineering knowledge formed in the software engineering ontology. We have analyzed instantiations in the software engineering ontology through the examples. Software engineering ontology

assists in defining information for the exchange of project information and is used in communicating across multisite development. Its end users are software engineers sharing domain knowledge as well as instance knowledge of software engineers.

Recent events such as the WOMSDE (Ontologies and Metamodeling in Software and Data Engineering) workshop, ONTOSE (Ontology, Conceptualization and Epistemology for Software and Systems Engineering) and SWESE (Semantic-Web Enabled Software Engineering) have focused on ontologies in software engineering, demonstrating that ontologies are becoming increasingly important in the area of software engineering

8. REFERENCES

[1] Software Engineering Ontology for Software Engineering Knowledge Management in Multi-site Software Development Environment , Pornpit Wongthongtham1, Elizabeth Chang1, Ian Sommerville

[2] Development of a Software Engineering Ontology for Multi-site Software Development Pornpit Wongthongtham, Elizabeth Chang, Tharam Dillon and Ian Sommerville

[3] Software Engineering Ontology : A development methodology , Olavo Mendes , Alain Abran

[4] IAENG International Journal of Computer Science, 33:1, IJCS_33_1_4 , ontologies and Object models in Object Oriented , Software Engineering Dr. Waralak V. Siricharoen

[5] Ontology Driven Software Engineering for Real Life Applications Michel Vanden Bossche1, Peter Ross2, Ian MacLarty2, Bert Van Nuffelen1, and Nikolay Pelov1

[6] Data modelling versus Ontology engineering Peter Spyns ,Robert Meersman ,Mustafa Jarrar

[7] Bourque, P. SWEBOK Guide Call for Reviewers. 2003 [cited 29 May 2003]; http://serl.cs.colorado.edu/~serl/seworld/database/3552. html.