An Ontology-based System to Support Distributed Software Development

Rodrigo G. C. Rocha, Ryan Azevedo, Eduardo Tavares, Daniel Figueredo
UFRPE
Garanhuns– PE, Brazil
rodrigo, ryan@uag.ufrpe.br,eteduardotavares,jdaniellI3593@gmail.com

Catarina Costa
Department of Statistics and Mathematics
Federal University of Acre
Rio Branco– AC, Brazil
catarina@ufac.br

Marcos Duarte
Information Systems Course
Paraíso College of Ceará
Juazeiro do Norte– CE, Brazil
marcos.duarte@fapce.edu.br

João Paulo Fechine
UNIPETECH
UNIPE
João Pessoa– PB, Brazil
fechine@gmail.com

Fred Freitas, Silvio Meira
Federal University of Pernambuco– CIn
Recife– PE, Brazil
fred,srlm@cin.ufpe.br

Abstract— **Distributed Software Development has become an option for software companies to expand their horizons and work with geographically dispersed teams, exploiting the advantages brought by this approach. However, this way of developing software enables new challenges to arise, such as the inexistence of a formal, normalized model of a project's data and artifacts accessible to all the individuals involved, which makes it harder for them to communicate, understand each other and what is specified on the project's artifacts. With that being said, this paper proposes a knowledge management tool that utilizes a domain-specific ontology for distributed development environments, aiming to help distributed teams overcome the challenges brought by this modality of software development proposing techniques and best practices. Thus, the main output of this work is Ontology-based System to Support the software development process with distributed teams.**

Keywords-Distributed Software Development; Ontologies; Knowledge.

I. INTRODUCTION

Motivated by opportunities like the availability of experts worldwide, cost reduction, local government incentives and employee turnover reduction, several software development companies have been starting to work with geographically distributed development teams, adopting the Distributed Software Development approach.

The aforementioned distribution of teams brings along with it new challenges to the software development scenario. Carmel [1] and Komi-Sirvo and Tihinen [2] reiterate the existence of these challenges by presenting some factors that are likely to lead distributed software development projects into failure: inefficient communication between distributed team members, diverging cultures and high complexity or lack of project management.

In this context, the nonexistence of a formal, normalized project data model accessible by the entirety of the team makes the communication between them and the understanding of the project artifacts harder, which can be aggravated when each member's culture and customs is barely or even not known by the rest of the team.

In order to mitigate these problems, the utilization of ontologies can be useful because they enable the creation of a common vocabulary. Wongthongtham *et al*. [3] mention that the use of ontologies represent a paradigm shift in Software Engineering and can be used especially to provide semantics for support tools, strong, knowledge-based communication, centralization and information availability.

This paper proposes DKDOnto, a domain-specific ontology for distributed software development projects, whose purpose is to aid those projects by defining a common vocabulary for distributed teams. Besides, this work proposes a tool that enables both handling and searching the information in the knowledge base, in order to get more useful information as to mitigate and avoid future problems inside the project.

The main goal of this work is the proposal of both the ontology and the tool, which together will compose a mechanism to ease the distributed software development process, from sharing of common knowledge between distributed team members or smart agents to the decision-making process effectuated by the project managers.

This paper is organized as follows: Ontology concepts are presented in Section II; Section III contains the knowledge-based system proposal; Related works are presented in Section IV, where a succinct analysis and comparison of related work and this paper is made; and, finally, Section V brings the final considerations.

II. ONTOLOGIES

Various definitions are given as to determine a meaning to ontologies in the Computer Science context, the most popular and best-known definition being "a formal, explicit specification of a shared conceptualization", given by Gruber [4]. By 'formal', he means that it is declaratively defined so that it can be comprehended by smart agents; by 'explicit', he means that the elements and their restrictions are clearly defined; by 'conceptualization', he means an abstract model of a field of knowledge or a limited universe of discourse; by 'shared', he indicates it is consensual knowledge, a common terminology of the modeled field. Thus, ontologies set an unambiguous, common higher abstraction level for several knowledge domains.

Ontologies, according to Guizzardi [5], are composed by concept, relations, function, axioms and instances. In short, concept can be 'anything' about 'something' that is going to be explained. The interaction between a domain's concepts and attributes is called relation, whose type is called function. Axioms model sentences that are always true and instances represent elements from the domain associated with specific concepts.

The use of ontologies has been made popular by many other Computer Science subfields, such as: Software Engineering, Artificial Intelligence, Database Design, and Information Systems. One of the principal persons responsible behind this phenomenon is Web Semantics' creator [6], Sir. Tim Berners Lee.

Many reasons instigate the development of ontologies, according to [7] [8]. Some of these reasons are:

- Sharing common understanding of how information is structured between humans and smart agents;
- Reusing knowledge of a domain. In case there is an ontology that adequately models certain knowledge of a domain, it can be shared and used by engineers and ontology developers, as well as teams that develop semantic and cognitive applications;
- Making explicit assumptions of a domain. Ontologies provide vocabulary to represent knowledge and its use prevents misinterpretations;
- Possibility of translation from and to various languages and knowledge representation formalisms. The translation concretizes an ideal pursued for generations by researchers in Artificial Intelligence. It makes it easier to reuse knowledge, and may allow for communication between agents in different formalisms, since this service is available in an increasing number of knowledge representation formalisms. Another way to reach this intent is to use ontology editors in which it is possible to choose in which language of representation the generated code is going to be written.
- The mapping between two knowledge representation formalisms, that, inspired in the connectivity component for Open Database Connectivity (ODBC) management systems, links two formalisms creating an common access interoperable interface for them, allowing an agent to access the other agent's knowledge.

Furthermore, ontologies help solve some of DSD project problems; for example, how to establish better communication, allow a homogenous comprehension of project information, make the project management a less laborious task, prevent task interpretation errors and synchronize the enrolled, distributed team's efforts and facilitate the knowledge sharing and standardization.

III. Knowledge-based System Proposal

In this work, we present the DKDOnto, a domain-ontology according to classification adopted by [9], which classifies the types of ontologies in: i) generic, ii) domain, iii) task and iv) application.

The ontology proposed intends to be the basis for possible solutions of knowledge-based systems in the context of global software development, in order to assist all the professionals (client too) involved in the software development process with distributed teams. The DKDOnto emerges, thus, as a common knowledge base for this context, leveraging the challenges deals, best practices and possible solutions, as well a road map with all the actors and their assignments.

This proposal takes a step beyond, discussing also an inference engine called DKDs, extremely flexible, customizable for each environment and giving support for the professional in real time. The general flow, operating means and features of the proposed system and the DKDOnto, as well as a systematic mapping study (methodology) are presented in the following subsections.

A. Systematic Mapping Study

In this research, a Systematic Mapping Study was conducted to identify ontologies supporting the DSD. And indirectly to identify tools, techniques, best practices, and models that use ontologies to support this area.

An important issue in this process was to search for reviews and accurate analyses on the field, looking for current researches and open challenges related to the use of ontological resources in Distributed Software Development processes. Thus, the following research question were intended to be answered: "Which ontologies have been proposed or adopted in the context of DSD?"

The searches for the primary studies were conducted according to the research plans defined in the protocol. The search process retrieved 1588 studies from the chosen scientific databases.

This question aims to find out which are the ontologies normalized on the DSD context. In order to answer this research question, four ontologies have been found. Table 1 presents the proposed ontologies in the distributed context. The first column presents the name and identifier of each ontology. The second column shows a description of each one.

Based on results, it is evident that the development phases that are benefiting from the use of ontologies are: process, management, requirements and design. On the other hand, some important branches have not been fully approached, for example, quality and tests, which involves lots of information management activities, and may have a considerable evolution with the utilization of ontologies as means to standardize, manage and share knowledge.

By answering the research question from this mapping, there have been found four works that propose some ontologies, especially developed for distributed software development, according to what was previously presented.

Since these ontologies have been designed specifically for distributed teams, they bear the concepts and features required to work in this environment. Noteworthy to mention that two of the four ontologies were developed for open-source software development communities. According to Mirbel [10], the free dynamic nature of this environment poses challenges to the coordination of activities and knowledge sharing.

Therefore, the use of ontologies as a support to open-source software development simplifies the management of knowledge resources in the communities. Noticeable that several other works use ontologies to solve or mitigate challenges and in DSD environments, however, these ontologies are not specific for this environment.

Thus, four ontologies have been found, they are not shared which does not allow further evaluation and according to the literature, they have not correct modeling to cover the entire software development process using distributed teams. But they have a major limitation, they have not resources to recommend best practices for possible problems.

There are numerous tools that utilize nonspecific-to-DSD ontologies only to mitigate challenges and limitations. These tools are distributed and used as support in the various project parts, from actual Software Engineering branches to specific project activities.

TABLE I. ONTOLOGIES FOR DSD

Models	Description
OFFLOSC[10]	This ontology is formalized in the context of open-source software development communities. Its goal is help coordinate activities, management of resources and knowledge sharing. It is composed by 46 classes and describes the concepts related to open-source communities such as actors, artifacts, activities, operations, relationships and resources.
Knowledge Management Ontologies [11]	A set of ontologies that formalize structural concepts of DSD environments, directed to knowledge management. It describes concepts of software artifacts, environment problems, interaction among the distributed development teammates, infrastructure, business rules and general information of the project.
Open Source Communities [12]	This ontology is also formalized in the context of open-source software development and its main purpose is to compose a project knowledge basis having semantically related, categorized data, which allows the execution of semantic searches and data inferences by smart agents. It is composed of 6 classes that describe concepts of actor's relations, rules, activities, processes, artifacts and tools from open-source communities' projects.
OntoDISEN [13]	This ontology is formalized in the DSD Project scenario and is used to aid the establishment of communication between distributed teams. It is integrated to a textual information-spreading model, enabling sharing information in distributed environments to be comprehended by all the software engineers in a clear, homogeneous way. It describes concepts of elements that are represented and shared in a DSD environment, such as users, tools, other environments, activities and processes.

With these results, it is clear that there are a lot of advantages in using ontologies to support DSD, especially to generate solutions aiming at mitigating the communication, collaboration, knowledge flow management, coordination of project activities and knowledge, and process management issues.

B. *DKDOnto: Proposal Ontology*

The DKDOnto ontology was developed using Ontology Engineering, Methontology [14] and IEEE Standard [15] for developing knowledge-based information systems methodologies; also, Method 101, proposed by N. F. Noy and D. L. Mcguinness's [7] was used a complement to Methontology.

Thus, the language used to build the ontology was OWL, which eases the publication and sharing of ontologies [16] and it has also been proposed as a standard for the World Wide Web Consortium (W3C), incorporating and taking advantage of the strength of earlier languages. OWL is an ontology language (Semantic Web [17]) with high-level expressivity and great potential for knowledge inference. In order to edit the ontology, the use of Protégé [18] was employed. It is a free, extensible, Java-based, open-source ontology editor and knowledge-based framework.

The DKDOnto has about 50 classes, but this paper describes the following core classes.

• Project: the main class of this knowledge base. It is responsible to store all the information about the settings of projects, from allocated team members to phases to activities to artifacts used.

• Member: it is a subclass of Resource. Member is an individual who has access to the environment and are allocated to Projects. A member has skills and works in a place and participates directly in the project, reporting best practices and challenges, using and creating artifacts.

• Best Practices: all the solutions and best practices used to face any problem should be stored in this entity. This class is responsible for helping avoid challenges and problems found and reported by a member during the execution of their activities. It also to solve these challenges and problems.

• Challenges: all the challenges and problems found by members should be stored in this class. A challenge can use best practices to solve itself. This entity is fundamental because the challenges has some solution or best practice associated with some practice can be used and available to another members with same problems.

• Skills: all members' knowledge are stored in this entity. The Member's skill enables to avoid challenges and solve it too. This class allows too that activities be distributed for the members according their skills.

• Place: it is a fundamental class to define exactly where the envolved member are in Project. This entity estores all information about member's localization, defining what is dispersion level and temporal distance.

• Artifact: class that is used by almost all other main classes. It supports members and their activities. Tools can use artifacts in specific activities, too.

• Tool: class reponsible for all the tools envolved in Project. It allows that all the users knows which tools are used for another members and another projects. This way, is possible to follow the patterns and find specific informations and instructions for use this tools.

• Workspace: is a class that contains Artifacts and Tools that Users can use and create in their activities. All the users allocated in that Project can be access the workspace for commit and checkout all the documents, artifacts or tools. The main goal of this class is storable of Artifacts and the Member uses the Workspace of that specific Project.

This ontology uses two fundamental classes for the sucess of this proposal. These classes are responsibles for storage all information about the problems and solutions during the project. These classes are called of Challenges and BestPractices. Thus, user's queries allows to view responses of the challenges, the knowledge base returns the best practices found for a certain team setting and can be applied to support challenges, which can be useful for other teams involved with the same project or other teams from different projects.

Figure 1 shows some relations between classes defined in DKDOnto. This diagram of generated from a plugin for Protégé called Ontoviz. For space constraints, a restrict set classes was chosen to be exhibited.

C. DKDs: Proposal Tool

DKDs was developed to aid in the transmission, generation and distribution of knowledge. It is a support tool for decision-making in DSD, which, based in resources and information from the context of a project, the system suggests possible solutions for the problems found to its users. In this sense, the system accesses the knowledge base having distributed projects experiences, their configurations, challenges faced and solutions used to overcome those challenges.

This tool’s main goal is to support the complete DSD process, offering recommendations considering the project setting and organization, technical and nontechnical experiences.

In order to develop DKDs, the general platform adopted was J2EE [19]; the web application frameworks utilized were Grails [20] (High-productivity web framework based on the Groovy language [21]) and Google Web Toolkit (GWT) [22]; Hibernate (Java persistence framework project) [23] was used for persistence; and to manipulate the ontology, the Jena framework was employed, which is also responsible for construction and manipulation of Resource Description Framework (RDF) [24] graphics.

With the DKDs a member from a project can know who are the another members envolved and have some instructions to talk each other depending their cultural characteristics. So, it helps to avoid any problems the communication (email, talk, phone). Furthermore, any doubt about some artifact or activity can be solved with the correct member, that is indicated by the tool.

Among DKDs’ main features, the most important ones are: DKDs uses the inference engine Pellet for inferring facts based on the information that has been previously stored in the knowledge base, thus, some outcomes that the system can generate:

- Starting the project, request a guideline with suggested best practices for similar contexts
- Starting the project, request a guideline with main challenges for similar contexts
- Determines who are the most qualified members to solve technical problems;
- Suggests possible practices, tools or techniques that can be employed to avoid challenges
- Find possible solutions used previously to problems encountered
- Evaluating the solutions proposed by the tool
- Suggest adaptations to the proposed solutions

The application is basically composed by four modules:

- Inference Module: allows for a precise deduction of information about DKDOnto in RDF and OWL code, using inference engine Pellet.
- Query Module: this is where all the queries made by users occur. As it was mentioned earlier, queries are made in SPARQL language and are transparent to the users.
- Views Module: gives access to all the reports made according to the users’ needs.
- Management Module: responsible for enabling access to the ontology with insertion, removal and editing of the data in the ontology permissions.

For example, an user can access the application and insert, delete, edit and view all the data (instances contained in DKDOnto) by the Management Module. The same user can use View Module for the ask the system to inform what is necessary, so, this module activates the Query Module that use the Inference Module to bring appropriate responses for the user.

The users have an access interface to execute the abovementioned functions on one side, whereas on the other side, there is the SPARQL (Query language for Resource Description Framework) [25] inference engine to consult

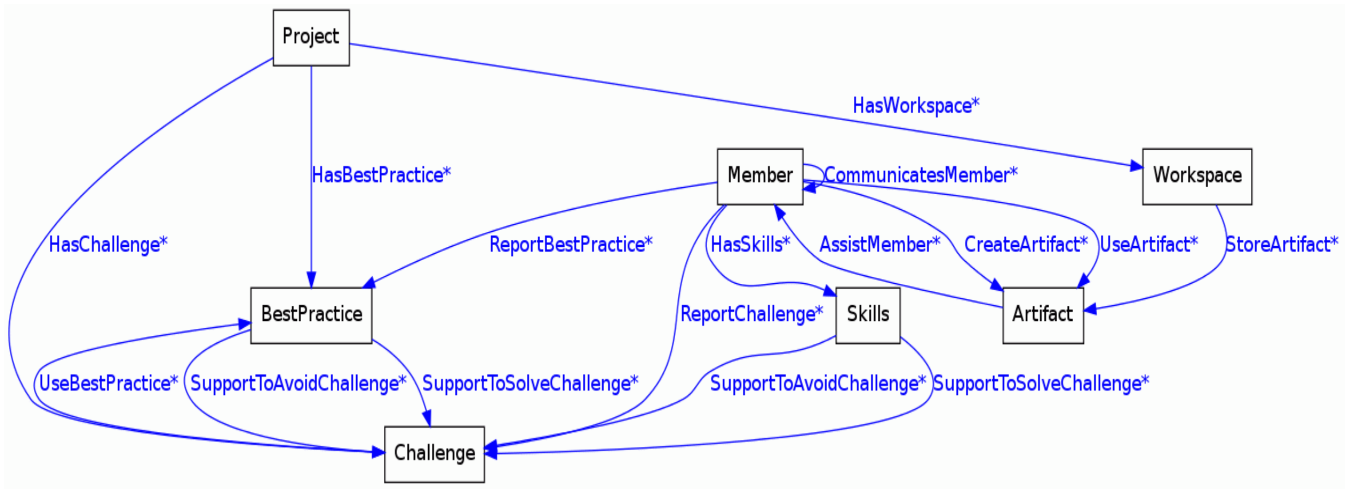

Figure 1. The Core classes and relationships of DKDOnto

DKDOnto, and the interface component (OWL API [26]) in the middle, which interacts with both sides. Integrating all the demands from user using the inference module.

Figure 2 shows the tool's general functioning as described above.

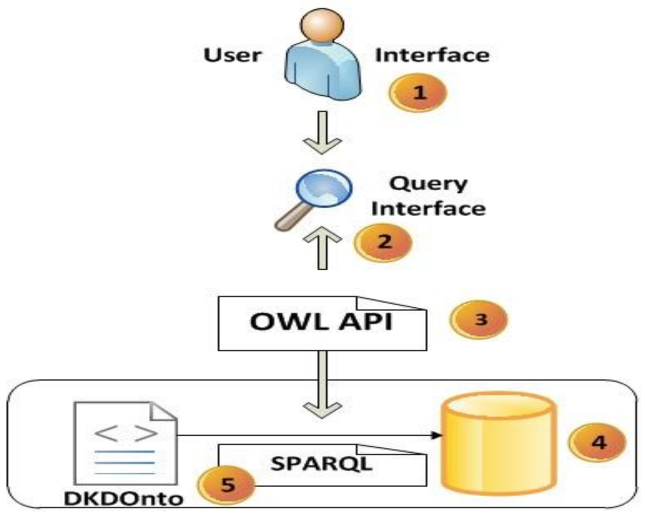

Figure 2. Tools General Functioning

IV. Related Work

In this section, works having the same goal or theme of this paper are described. Based on the amount of related works found, it can be affirmed that relatively few works on Software Engineering Ontologies have been carried out.

Wongthongtham *et al.* [27] present the project and implementation of a social network approach as a mean to support the sharing and evolution of a Software Engineering ontology. A multi-agent recommender system that uses the 'Software Engineering Ontology' and 'SoftWare Engineering Body of Knowledge (SWEBOK)' as sources of knowledge is designed within multi-site communities of software engineers and developers working on related projects as the target audience. Though a big challenge faced by this approach is ensuring that the knowledge bases of different agents are coherent and consistent with one another, as stated by Dilon and Simmons [28].

Ankolekar *et al.* [29] considers as one of the toughest problems faced by online professional communities the fact that the vast amount of data generated as a result of their interactions is not well-linked on the basis of the meaning of its content. With the assumption that a better semantic support can bring improvements to these communities, a prototype Semantic Web system was developed.

Such task required a way of describing the semantic content retrieved from the data obtained from these communities, which was accomplished through the use of ontologies. The large amount of data generated was a large obstacle as the parsers used were unable to reason efficiently for large amounts of data.

The 'instance Store (iS)' system was the solution for such problem, for it stores assertions about individuals and their types in a database, reducing reasoning over individuals to terminological reasoning. But the version of iS used was limited to role-free reasoning of individuals, what at first was deemed to be a major limitation but was dismissed by the authors since the primary use of ontologies in the system "is for the description, annotation and retrieval of large number of individuals" and it "does not make use of the open world assumption nor does it make use of ontologies distributed over multiple sites".

In their work, Dillon and Simmons [28] reiterate the growing importance of the use of ontologies in various aspects of Software Engineering, showing examples ranging from the support that offered to multi-site developers, to the provision of semantics to different categories of software. The 'Software Engineering Ontology' is described and used for the creation of a software engineering knowledge management system that is formed by a 'safeguard system', 'ontology system' and a 'decision-maker system'. The purpose behind this system is to facilitate knowledge sharing, access, update and exchange.

The essential difference of this work is the proposal of the use of best practices for the challenges found by any member, thus, they can be use the DKDs to check or consult all knowledge stored looking for possible best practices. It also allows the creation of a list of possible problems during the initial phases, so the manager or developers can avoid some challenges. Other interesting resource is the creation of a list of possible developers who may be able to help solve technical problems through their skills.

V. Conclusion

As globalization took place, the distribution of software development processes have become an increasingly common fact. The DSD work environments are very complex and there are no mature practices for this context since it is relatively new. In this sense, ontologies can bring benefits such as a shared understanding of information, ease of communication among distributed teams and effectiveness in information management.

This work presents evidences from collected papers and a briew analysis of the results reached. The results support the foundation for proposing and developing a feature based on ontologies to support the DSD. The systematic mapping aimed to identify ontologies formalized in DSD context, provided that advance the state of art, highlighting the need to use ontology in this field. Is possible to view all the Systematic Mapping Results in Borge's work [30]. The complete information about it is available at a specific repository files [31].

DKDOnto and DKDs fulfill what has been proposed, consisting of a computing tool that can be used for treatment, analysis and utilization of information on distributed software projects. In this sense, the ontology and the tool allow that actors in this scenario obtain and access correct information and artifacts, providing a high-level knowledge model for the team members.

The results obtained to this date are expressive, in which, for example, the project manager has actual consistent knowledge of which cultures are involved in the distributed teams and which are the implication of this, which enables them to handle each case effectively. Similarly, a technical leader has access to the project participants' technical knowledge, making them able to require or assign specific activities accordingly to the expertise of each team member.

Another important point is that the ontology, as presented in Section 3, has two fundamental classes, namely Challenges and Solutions that are directly utilized by the query tool. That way, the knowledge base will return the challenges found for a certain team setting and also which solutions can be applied to such challenges, which can be useful for other teams involved with the same project or other teams from different projects.

The next step in this segment is to concretize the acquisition of knowledge in a systematic way in order to fill the ontology. In this case, it will be possible to make tests and simulations with higher precision since all the inserted data will be from real projects. Furthermore, other techniques can be used for improves the support of Challenges, for example, the use of natural procesing language for retrieve better solutions or best practices based in challenges cases.

ACKNOWLEDGMENT

Rodrigo was supported partially by CNPQ and Ryan are supported by CNPq. This work is partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

REFERENCES

[1] E. Carmel. "Global Software Teams: Collaboration Across Borders and Time Zones". Prentice-Hall, EUA. 1999.

[2] S. Komi-Sirvo and M. Tihinen. "Lessons Learned by Participants of Distributed Software Development". Journal Knowledge and Process Management, vol. 12 no 2, 2005, pp. 108–122.

[3] P. Wongthongtham, E. Chang, T. Dillon, and I. Sommerville. "Ontology-based Multi-site Software Development Methodology and Tools". J. of Systems Architecture. ACM, New York. 2006. 640–653.

[4] T. Gruber. "Toward Principles for the Design of Ontologies used for Knowledge Sharing". In formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers. 1995.

[5] G. Guizzardi. "A methodological approach to development and reuse, based on formal domain ontologies" Master Degree. Federal University of Espírito Santo. 2000.

[6] T. Berners-Lee, O. Lassila, and J. Hendler. "The semantic web." Scientific American, 2001, pp. 5:34–5:43.

[7] N. Noy and D. Mcguinness,. "Ontology development 101: A guide to creating your first ontology,". [Online].Available:http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html. [retrieved: 06, 2013]. 2001.

[8] F. Freitas. "Ontologies and the semantic web". Proceedings of XXIII Computer Sience Brazilian Society Symposium. Campinas: SBC. v. 8, 2003, pp. 1-52.

[9] N. Guarino, "Formal ontology and information systems," in Proceedings of FOIS98. Trento, Italia: IOS Press, pp. 3–15. 1998.

[10] I. Mirbel. "OFLOSSC, "An Ontology for Supporting Open Source Development Communities". In Proceedings of the International Conference on Enterprise Information Systems (ICEIS). 2009.

[11] W. Maalej and H. Happel. "A Lightweight Approach for Knowledge Sharing in Distributed Software Teams". In Proceedings of the Practical Aspects of Knowledge Management (PAKM). 2008.

[12] T. Dillon and G. Simmons. "Semantic Web support for Open-source Software Development". In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS). 2008.

[13] A. Chaves, I. Steinmacher, C. Lapasini, E. Huzita, and A. Biasão. "OntoDISEN: an Ontology to Support Global Software Development". CLEI Electronic Journal. 2011. v. 14, pp. 1-12.

[14] M. Fernandez, A. Gomez-Perez, and N. Juristo, "Methontology: from ontological art towards ontological engineering," in Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA, 1997, pp.33–40.

[15] IEEE, "Standard for developing software life cycle processes". p. 96, may 1997, eEE Computing Society. Available: http://standards.ieee.org/catalog/olis/archse.html.[retrieved:06, 2013]. 1997.

[16] OWL. Web ontology language overview. Available: http://www.w3.org/TR/owl-features.[retrieved: 06, 2013]. 2009.

[17] J. Berners-Lee and O. Lassila, "The semantic web," Scientific American Magazine.[retrieved: 06, 2013]. 2001.

[18] Protégé. Protégé ontology editor. Online. [Online]. Available: http://protege.stanford.edu/doc/users.html. [retrieved: 06, 2013]. 2009.

[19] J2EE. JAVA Enterprise Edition. Available: http://oracle.com/technetwork/java/javaee/overview/index.html, [retrieved: 06, 2013]. 2013.

[20] Grails. Available: http://grails.org, [retrieved: 06, 2013]. 2013.

[21] Groovy. Available: http://groovy.codehaus.org, [retrieved: 06, 2013]. 2013.

[22] Google Web Toolkit. Available: http://gwtproject.org, [retrieved: 06, 2013]. 2013.

[23] Hibernate. Available: http://hibernate.org, [retrieved: 06, 2013]. 2013.

[24] J. Carroll, D. Reynolds, I. Dickinson, A. Seaborne, C. Dollin, and K. Wilkinson, "Jena: Implementing the semantic web recommendations" . pp. 74–83. 2004.

[25] SPARQL Query Language for RDF. Available: http://w3.org/TR/rdf-sparql-query, [retrieved: 06, 2013]. 2013.

[26] OWL API. Available: http://owlapi.sourceforge.net, [retrieved: 06, 2013]. 2013.

[27] P. Wongthongtham, E. Chang, and T. Dillon. "Ontology-based Multi-agent System to Multi-site Software Development". In Proceedings of the Workshop on Quantitative Techniques for Software Agile Process (QUTE-SWAP). (Newport Beach, USA). 2004.

[28] T. Dillon, G and Simmons, G. "Semantic Web support for Open-source Software Development". In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS). 2008.

[29] A. Ankolekar, K. Sycara, J. Herbsleb, and R. Kraut. Welty Chris. Internactional Conference on World Wide Web. Pg 575-584. 2006.

[30] A. Borges, R. Rocha, C. Costa, H. Tomaz, S. Soares, and S. Meira. "Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study". In Proceedings of the International Conference on Evaluation & Assessment in Software Engineering (EASE). Porto de Galinhas, PE, Brasil. 2013.

[31] Files Repository of Mapping Study about Ontologies in Distributed Software Development: http://www.rgcrocha.com/ms, [retrieved: 06, 2013]. 2013.