DKDs: An Ontology-based System for Distributed Teams

Rodrigo G. C. Rocha, Ryan Azevedo, Dimas Cassimiro, Ana Raquel Morais
Federal Rural University of Pernambuco - UFRPE
Garanhuns, Brazil
rodrigo, ryan, dimas@uag.ufrpe.br, anaraqueldemorais@gmail.com

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

Silvio Meira
Center of Informatics,
Federal University of Pernambuco (CIn-UFPE)
Recife, PE, Brazil
srlm@cin.ufpe.br

Abstract— Global 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. Then, 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—Global 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.

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, is presented 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 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 DKD, 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.

A. 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 involved member are in Project. This entity stores 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.

This ontology uses two fundamental classes for the sucess of this proposal. These classes are responsible 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.

Another important concept about ontologies is the Axiom. Freitas [8] affirms that Axioms are an important component of ontologies to describe the relationships among the concepts. The DKDOnto used a set of axioms to ensure the rules were met by knowledge domain. For a better contextualization, some axiom are presents bellow, in Figure 1.

![Axioms from DKDOnto](image)

All the main concepts used in Axioms are refers to the fundamental classes from DKDOnto. The first means that a Project has a Bestpractice, so, the second means that Bestpractice can solve a Challenge, then the manager can authorize Challenges related by another members, Manager is a Member, Member executa the activities or tasks, and finally, Member user Tools. It was verified the creation of an existential quantifier in generated axioms indicating that Description Logic was used.

B. 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, the Inference Module: allows for a precise deduction of information about DKDOnto in RDF and OWL code, using inference engine Pellet. The 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. The Views Module: gives access to all the reports made according to the users’ needs. And finally, the 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 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.

![Tools General Functioning](image)
IV. 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.

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. This work was realized after a systematic mapping that 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 [27].

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 processing language for retrieve better solutions or best practices based in challenges cases.

REFERENCES