Cordis-FBC: an Enterprise-Oriented Software Development Environment

Karina Villela [1,2], Kathia Oliveira[3], Gleison Santos [1]
Ana Regina Rocha[1], Guilherme Travassos [1]

[1] Federal University of Rio de Janeiro – COPPE
Caixa Postal 68511 – CEP 21941-972 – Rio de Janeiro, RJ - Brazil
{kvillela, gleison, ght, darocha}@cos.ufrj.br

[2] Federal University of Bahia
Unit of Cardiology and Cardiovascular Surgery/Fundação Bahiana de Cardiologia
Rua Augusto Viana s/n – Canela
CEP 40140-060 – Salvador, BA - Brazil

Catholic University of Brasilia
SGAN 916, módulo B Asa Norte
CEP 70790-160 - Brasilia, DF - Brazil
kathia@ucb.br

Abstract: Knowledge has been thought to be the most important asset in a company having a significant impact on company competitiveness. Software development is knowledge-intensive but software development environments lack specific support of knowledge management. In this paper, an approach is presented to give access to specific and valuable knowledge during software development with an Enterprise-Oriented Software Development Environment.

1 Introduction

Knowledge has been thought to be the most important asset in a company having a significant impact on company competitiveness. Nevertheless, to accomplish competitive advantage, knowledge must not remain at the individual level. Knowledge at the individual level is lost when individuals leave the company and it is often difficult to find and access it. The organizational knowledge recorded in documents also represents a problem when it cannot be easily accessed, shared and updated.

Software development is knowledge-intensive. Several knowledge representations and transformations are required along the software development process, and different kinds of knowledge are important to software developers in this context, among which are: domain knowledge, organizational guidelines, best practices and previous experiences with techniques, methods and the software process.

In this paper, we present the Domain-Oriented Software Development Environment (DOSDE) Cordis and its evolution to the Enterprise-Oriented Software Development

Environment (EOSDE) Cordis-FBC. These environments are part of a long-term project between the Federal University of Rio de Janeiro and the Unit of Cardiology and Cardiovascular Surgery/ Fundação Bahiana de Cardiologia (UCCV/FBC) of the Federal University of Bahia.

After some years of partnership, we realized that the crucial problems of software development at UCCV/FBC were the lack of cardiological knowledge, in addition to the software teams' lack of knowledge about the organization itself. In an evolutionary improvement approach, we dealt first with the lack of cardiological knowledge, which led to the DOSDE Cordis. As we have observed that this approach was able to improve software practices but was not complete, the next step was to improve the software development environment provided in order to deal with organizational knowledge. This led to the EOSDE Cordis-FBC.

2. Cordis, a Domain-Oriented Software Development Environment

The major characteristics of software projects carried out at UCCV/FBC are that they involve research and are usually developed by small groups. The characteristics of these working groups have changed over time, but an important aspect, the high turnover of developers, is always present due to the projects being sponsored by temporary grants from the Brazilian government. To address this problem we have defined the DOSDE Cordis for the cardiology domain. Cordis supports the development of software in Cardiology by considering the embedded knowledge of this domain to guide the software developers across the several phases of the software process.

To support these features we have first defined Domain Theory as the domain model that contains the knowledge to assist the software developers. A Domain Theory uses a domain ontology [UG96] and contains an identification of tasks mapped with the domain knowledge. These tasks represent activities that take place in the specific domain (e.g. diagnosis). The Domain Theory can be divided in sub-theories. The following sub-theories were defined for cardiology: *(i)* heart anatomy (concepts about the heart structure and the physiology), *(ii)* findings (concepts that are used in the physician's investigation process), *(iii)* therapy (general kinds of therapies and their features), *(iv)* diagnosis (concepts and characteristics that identify syndrome and etiological diagnoses); and *(v)* pathologies (representing different situations of heart components whose classifications and features are nevertheless important for the purpose of the software development). The tasks identified for cardiology are: diagnosis, therapeutic planning, simulation and monitoring. The mapping provides an idea of the concepts more closely related with the task and those to which the software developers should give special attention.

The next step, therefore, was the implementation of this approach. We have used the infrastructure of the TABA workstation, a meta-environment developed at COPPE/UFRJ and capable of generating, through instancing, process-centered software development environments. We extended the TABA infrastructure to allow the definition of specific development processes starting from a standard process, and to include components called Domain Theories and the required tools to handle them [Ol99], [Vi02]. After the

definition of the domain theory and the software process, the next step was to integrate domain specific tools. These tools differ from more classical ones because they use the vocabulary of a specific domain. We built, for instance, a knowledge editor to assist in knowledge acquisition considering the cardiology domain [Ol00].

3. Cordis-FBC, an Enterprise-Oriented Software Development Environment

Besides domain knowledge, we realized that other kinds of knowledge would be of great interest. They include knowledge about the organization itself and data and experience obtained on previous software development projects within the organization. For the development of some kinds of medical software, the knowledge about the organizational processes at UCCV/FBC is very important, even essential. Due to the high turnover within the software teams, medical personnel often complained that they had to explain a process, once again, to a recently-arrived developer. Another aspect observed was the importance of identifying key personnel in the organization who has the specific knowledge for an activity to be carried out during a software project. This information can ensure that the right people can be contacted, saving valuable time in the information gathering process. As a consequence, we decided to have an organizational model, looking at its structure, its processes, and the distribution of knowledge and skills throughout this structure and these processes. Besides the availability of an organizational model we start to register lessons learned throughout the software processes in order to derive the best practices for the organization.

This was the motivation for broadening DOSDE and defining EOSDE, which has the following goals: (a) to provide software developers with all relevant knowledge for software development accumulated by the company, and (b) to support organizational learning about software development. Now we will describe the requirements for an EOSDE, its knowledge infrastructure and the extension made on TABA Workstation.

3.1 Requirements for an EOSDE

In order to achieve its goals, EOSDE must meet the following requirements: (i) provide the representation of the organizational structure and processes, and make it easy to find experts; (ii) store specialized knowledge and supply this knowledge to a project team when necessary; and (iii) support the continuous evolution of knowledge in the environment.

If a project team has access to the organizational structure and to the staff assigned to it, it is easier to find the experts who can contribute to their project. The representation of organizational processes both enables a better understanding of these processes and of the role of an expert in the company and provides the context for the use of a skill or knowledge. These are the reasons for including requirement (i). In order to address requirement (ii), EOSDE should include knowledge about the software process of the company. Moreover, EOSDE should contain knowledge about the experience on software development acquired by the company. Knowledge and experience evolve with

time and it is important to support the evolution of knowledge in EOSDE (requirement (iii)).

3.2 Knowledge Infrastructure for EOSDE

The use of ontologies in the EOSDE knowledge infrastructure is critical to make easier the communication between multiple users and the retrieval of knowledge stored in the environment. Considering communication, the defined ontologies have the purpose of reducing terminological and conceptual confusion in a company. As to the retrieval of knowledge items, the purpose of the ontologies is to supply vocabularies whose terms are used as links among the contents of the multiple knowledge/data bases. Besides, when defining synonyms and acronyms for the concepts, the ontologies provide linguistic equivalents which may occur in text documents and may be used to classify and access informal knowledge.

The components of knowledge infrastructure are described below:

- *Software Engineering (SE) Theory* component: This component contains the *SE Ontology* component, which defines a common vocabulary to guide the registration/distribution of a company's knowledge map and of software engineering knowledge by an EOSDE. A company's knowledge map defines which skills, knowledge and experiences each employee has and to which degree these skills, knowledge and experiences are held. When the knowledge map is being recorded or updated, the knowledge about Software Engineering that a certain employee has is defined based on the terms of *SE Ontology*. The software engineering knowledge stored in the environment is also associated to *SE Ontology* terms. Each knowledge item is associated to one or more terms, which enables subsequent retrieval of different types of knowledge items based on *SE Ontology* terms, independently from the specific tool used to obtain or to record the knowledge items.
- *Task Description* component: This component contains the description of generic tasks, that is, tasks applicable to different domains. A task description consists of a high-level description, a task ontology and bibliographic references. A task is a sequence of necessary steps for the solution of a problem, so a task is also described in terms of its sub-tasks. A *Task Ontology* contains concepts and attributes associated with a generic task. The objective of describing generic tasks is to obtain knowledge, regardless of domain, which helps software developers understand a problem through the understanding of the tasks that make it up. A model for task ontologies for EOSDE was defined in [ZOR02].
- *Company Description* component: This component contains the description of a company, identifying the generic tasks that are performed and the software engineering knowledge necessary in the context of the organizational structure and processes. Hence organizational processes and the activities of these processes can be mapped for generic tasks and the description of a task can be used to help the company's knowledge manager to define an organizational process. Software engineering concepts are mapped both for organizational process activities and for the positions of the organizational structure in which they are required, enabling a better understanding of organizational processes and structure and also of the role of knowledge in the company. The company's knowledge map is also part of the

Company Description component. The main component of *Company Description* is the *Company Ontology*, which provides concepts and attributes related to the structural, behavioral and knowledge aspects of companies, defining a common vocabulary to guide the description of any company. The *Company Ontology* was developed combining new concepts with the concepts defined by the Enterprise Ontology [Us98] and by the TOVE's (TOronto Virtual Enterprise) ontology [FBG96].

- *Domain Theory component:* This component organizes domain knowledge by using *Domain Ontology* as described in section 2.

The remaining components are *Knowledge and Data Bases* and *Knowledge Management Services*. *Knowledge and Data Bases* store the knowledge and data acquired through many software projects. *Knowledge Management Services* allow the storage of knowledge in the organizational memory and the dissemination and evolution of stored knowledge. Some of these components are mentioned in [Ab98], [Al99].

3.3 Extension of TABA Workskstation

Once again the infrastructure of the TABA workstation was extended and new tools were defined. Now the TABA workstation allows to configure an Enterprise-Oriented Software Development Environment. An EOSDE include knowledge accumulated by the organization that has been captured for its importance for software development. This includes the organizational model, lessons learned and the best practices identified over a period of time. Starting from the configured environment, specific environments can be instantiated for individual projects taking into account their characteristics (size, complexity, schedule etc.). All these environments have access to the knowledge available at the configured EOSDE. Cordis-FBC was the first EOSDE configured according to this approach.

4. Conclusion

In this paper we described an approach to support Knowledge Management throughout the execution of software processes. Our goal was to provide the development team with domain knowledge, knowledge about the organization itself and knowledge about the most successful software practices in the organization.

We are aware of the difficulty of capturing and modeling domain knowledge. Modeling and continuously updating the organizational structure model is also time consuming, but it may benefit the whole organization as it allows inadequacies on the structure, on the processes or on the distribution of knowledge, skills and experience to be detected and fixed. The benefits of the availability of lessons learned and best practices within the organization are well known. Therefore their acquisition and dissemination need a well established policy of incentives to be successful.

The contribution of this work, when compared to related ones [MS01, VLV01], consists of the explicit introduction of knowledge about a company in its software development

environment. It is worth highlighting the supply of tools for specific activities, which provide related organizational knowledge even if an explicit request has not been made. Both the role played by domain and organizational knowledge on software development and maintenance and its effectiveness are going to be evaluated through the use of Cordis-FBC and its instantiated environments for individual projects.

Acknowledgments

We acknowledge CNPq for its financial support.

References

[Ab98] Abecker, A. et al.: Toward a Technology for Organizational Memories, IEEE Intelligent Systems, v. 13, n. 3, 1998; pp. 40-48.

[Al99] Althoff, K.. et al.: Managing Software Engineering Experience for Comprehensive Reuse. In: Proc. 11th International Conference on SE&KE, Kaiserslautern, 1999, pp.10-19.

[FBG96] Fox, M., Barbuceanu, M., Gruninger, M.: An Organization Ontology for Enterprise Modeling: Preliminary Concepts for Linking Structure and Behaviour, Computers in Industry, v. 29, 1996, pp. 123-134.

[IS95] ISO/IEC 12207*:* Information technology - Software Life Cycle Processes, International Standard Organization, 1995.

[MS01] Mendonça, M., Seaman, C.: A Prototype Experience Management System for a Software Consulting Organization. In: Proc. 13th International Conference on SE&KE, Buenos Aires, 2001, pp. 29-36.

[Ol00] Oliveira, K. et al.: A Generic Architecture for Knowledge Acquisition Tools in Cardiology. In: Proc. Workshop on Intelligent Data Analysis in Medicine and Pharmacology – 14th European Conference on Artificial Intelligence, Berlin, 2000; pp. 43-45.

[Ol99] Oliveira, K. et. al.: Using Domain-Knowledge in Software Development Environments. In: Proc.. 11th International Conference on SE&KE, Kaiserlauthern, 1999; pp. 180-187.

[UG96] Uschold M, Gruninger M.: Ontologies: principles, method and applications, The Knowledge Engineering Review, 1996, 11(2); pp. 93-136

[Us98] Uschold, M. et al.: The Enterprise Ontology, The Knowledge Engineering Review, v.13, 1998; In: www.aiai.ed.ac.uk/project/enterprise/ enterprise/ontology.html.

[Vi02] Villela,K. et al. :The Definition and Automated Support of Software Processs, taking Domain Knowledge and Organizational Culture into Consideration. In: Proc. Workshop on Software Quality – International Conference on Software Engineering, Orlando, 2002: pp. 8-11

[VLV01] Von Wangenheim, C. Lichtnow, D., Von Wangenheim, C.: A Hybrid Approach for Corporate Memory Management Systems in Software R&D Organizations. In: 13th International Conference on SE&KE, Buenos Aires, 2001, pp. 326-330.

[ZOR02] Zlot, F. Oliveira,K., Rocha, A.R., Modelling Task Knowledge to Support Software Development. In: Proc. 14th International Conference on SE&KE (SEKE'2002), Ischia, 2002, pp. 35-42.