Integrating Knowledge Management and Groupware in a Software Development Environment

Ricardo A. Falbo, Daniel O. Arantes, Ana C.C. Natali

Computer Science Department, Federal University of Espírito Santo, Vitória - ES - Brazil
falbo@inf.ufes.br

Abstract. Knowledge is one of the organization's most valuable assets. In the context of software development, knowledge management can be used to capture knowledge and experience generated during the software process. In this process, collaboration technologies also play a central role. With groupware facilities, we have the basis for creating, increasing and capturing knowledge from group and organizational collaboration. First in this paper, we discuss the importance of knowledge management in software development, and how knowledge management and groupware facilities can be integrated into a software development environment (SDE). Then, we present an infrastructure to manage knowledge in ODE, an Ontology-based software Development Environment. This infrastructure deals with several knowledge items, including artifacts, lessons learned and packages of discussion, and considers knowledge capture, store, retrieval, dissemination, use and maintenance.

1 Introduction

Software development is a collective, complex, and creative effort. In order to produce quality software, software organizations are trying to better use one of its most important resource: the organizational software engineering knowledge.

Historically, this knowledge has been stored on paper or in people's mind. When a problem arises, we look for experts across our work, relying on people we know, or we look for documents. Unfortunately, paper has limited accessibility and it is difficult to update [1]. In the other hand, in a large organization, it can be difficult to localize who knows some matter, and knowledge in people's mind is lost when individuals leave the company. Important discussions are lost because they are not adequately recorded. Therefore, knowledge has to be systematically collected, stored in a corporate memory, and shared across the organization [2]. In other words, knowledge management is necessary.

Knowledge management (KM) can be viewed as the development and leveraging of organizational knowledge to increase an organization's value [3]. KM involves human resource, enterprise organization and culture, as well as the information technology, methods and tools that support and enable it [4]. A KM system facilitates creation, access and reuse of knowledge, and its main goals are to promote knowledge growth, communication, preservation and sharing.

In the context of software development, KM can be used to capture the knowledge and experience generated during the software process. Reusing knowledge can prevent the repetition of past failures and guide the solution of recurrent problems. Also, we cannot forget that collaboration is one of the most important knowledge sources for software organizations. Then KM should be integrated with groupware applications. But, to be effective in the software development context, a KM system should be integrated to the software process. Since Software Development Environments (SDEs) can be viewed as software process automation, integrating collections of tools to support software engineering activities across the software lifecycle [5], it is natural to integrate groupware and KM facilities into a SDE.

This paper presents the KM infrastructure developed for ODE, an ontology-based SDE [6], and it is organized as follows: section 2 discusses the symbiosis between KM, groupware and SDEs; section 3 presents ODE's KM infrastructure; section 4 discusses related works, and, finally, in section 5, we report our conclusions.

2 KM, Groupware and Software Development Environments

Since nowadays knowledge is acknowledged as one of the most valuable organization's assets, it is important to manage organizational knowledge. KM combines tools and technologies to provide support to the capture, access, reuse and dissemination of knowledge, generating benefits for the organization and their members. In this context, collaboration technologies play a central role. KM applications form a continuum from low to high interaction complexity. Therefore, sharing knowledge requires using different interactive communication modes, according to the degree of shared contextual knowledge. In cases where knowledge can be explicitly encoded and recorded, or where the context is well-shared, collaboration technologies are useful in knowledge acquisition, combination, interpretation, and dissemination. Where knowledge is primarily tacit, these technologies can be used to support the personal interaction required for knowledge sharing, creation, and explication [3]. In fact, knowledge utilization and transfer cannot succeed without effectively supporting collaboration. On the other hand, collaborative problem solving, conversations, and teamwork generate a significant part of the knowledge assets that exist in an organization [7].

In the software development context, KM and groupware are quite essential. Using a KM approach, knowledge created during software processes can be captured, stored, disseminated, and reused. KM can be used to better support several activities, such as software process definition, human resource allocation, estimation, requirement analysis, quality planning, and so on.

Also, developing software is essentially a cooperative task. Developers act jointly to achieve the goal of producing a quality software product, and groupware applications, such as email, chat and forum, can be used to support interactions. The knowledge embedded in those interactions can also be captured, stored, disseminated, and reused. Thus, KM and groupware should be integrated. In fact, both KM and groupware should be integrated to the organizational process [8], which, in the case of software development, is the software process.

To deal with complex software processes, it becomes essential to provide computer-based tools to support software engineers to perform their tasks, and those tools must be integrated in a Software Development Environment (SDE). Consequently, SDEs, KM and groupware complement each other in supporting developers during the software process to produce better quality software.

In the next section, we discuss how KM and groupware facilities are integrated in ODE, an ontology-based software development environment.

3 Knowledge Management and Groupware in ODE

ODE (Ontology-based Development Environment) [6] is a process-centered SDE, that is developed grounded on ontologies. ODE's design premise is based on the following argument: if the tools in a SEE are built based on ontologies, tool integration can be improved. The same ontology can be used for building different tools supporting related software engineering activities. Moreover, if the ontologies are integrated, integration of tools built based on them can be highly facilitated [6].

ODE is developed based on some software engineering ontologies, such as an ontology of software process [9], an ontology of software metrics [10], and an ontology of software risks [11]. The most important of them is the software process ontology, since it describes the main concepts involved in software processes, such as process, project, activity, artifact, resource, procedure, and so on. The others ontologies are integrated to it, forming a net of concepts.

ODE is developed at the Software Engineering Laboratory of the Federal University of Espírito Santo, and it is implemented in Java. ODE has several tools developed based on its ontologies, such as tools supporting software process definition [6], estimation, resource allocation, risk analysis [11], quality control [6], documentation, and object modeling, among others. Figure 1 shows the software process definition tool. Looking to its interface, we can see that it is strongly based on the software process ontology, using its concepts, relations and constraints.

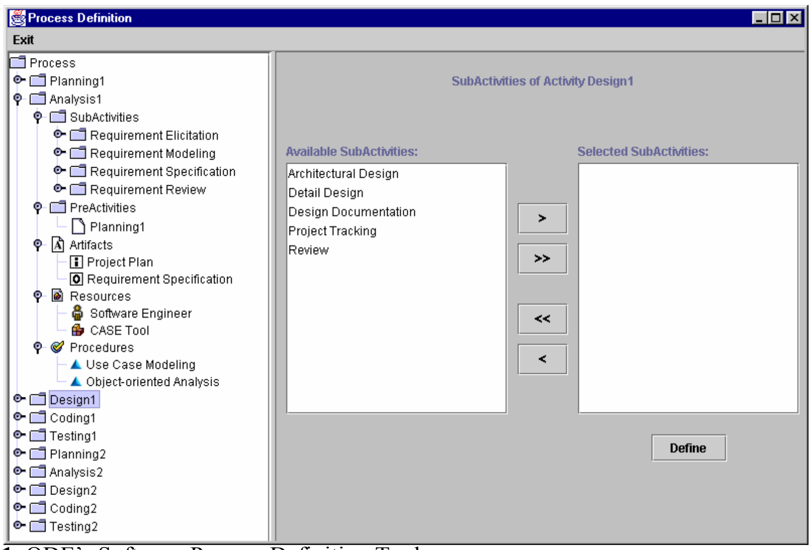


Fig. 1. ODE's Software Process Definition Tool.

As pointed in section 2, KM can be used to better support several activities, and consequently several tools in a SDE can be improved by some KM facilities. To support developing such tool's KM facilities, a KM infrastructure was developed for ODE. This infrastructure is organized as shown in Figure 2. The *organizational memory* (OM) is at the core of the infrastructure, supporting knowledge sharing and reuse. Arranged around the OM, KM services are available, supporting activities of a general KM process, including creation, capture, retrieval, access, dissemination, use, and preservation of organizational knowledge.

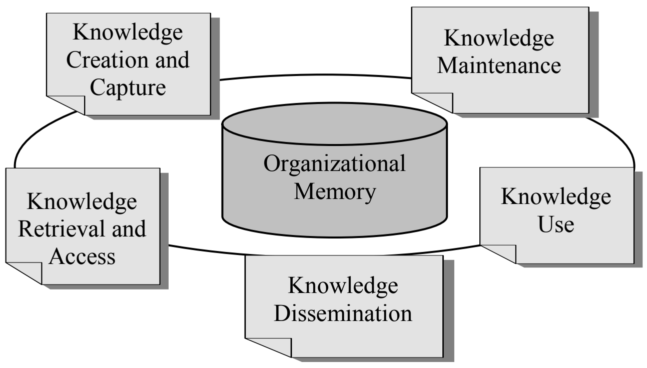


Fig. 2. ODE's KM Infrastructure.

Ontologies are particularly important for KM. As pointed by Staab et al. [12], ontologies constitute the glue that binds KM activities together, allowing a content-oriented view of KM. Ontologies define the shared vocabulary used in the KM system, facilitating communication, integration, search, storage and representation of knowledge. In ODE's KM infrastructure, ontologies are used to structure the OM, as well as to support some knowledge services, such as search and dissemination. Following, the OM structure and the KM services of ODE's KM infrastructure are discussed in more details.

3.1 ODE's Organizational Memory

As pointed above, ODE's OM is structured based on ontologies. As shown in Figure 3, ODE's OM is composed of several knowledge repositories, which are grounded on at least one ontology. Knowledge repositories, in turn, store several types of knowledge items that are relevant to software development, including artifacts, lessons learned, and message packages. Also knowledge repositories contain ontology instances (treated as Knowledge), that are created by the knowledge manager. These are used to store general knowledge about the software engineering domain described through ODE's software engineering ontologies, and are used for indexing knowledge items. In fact, the ontology instances can be viewed as a type of knowledge item, since they are formally defined from ontologies, and they are used in several situations in ODE, such as for giving suggestions during process definition (figure 1). However, we decided not to threat them as knowledge items, because there are some KM services, such as knowledge use, that do not apply to them.

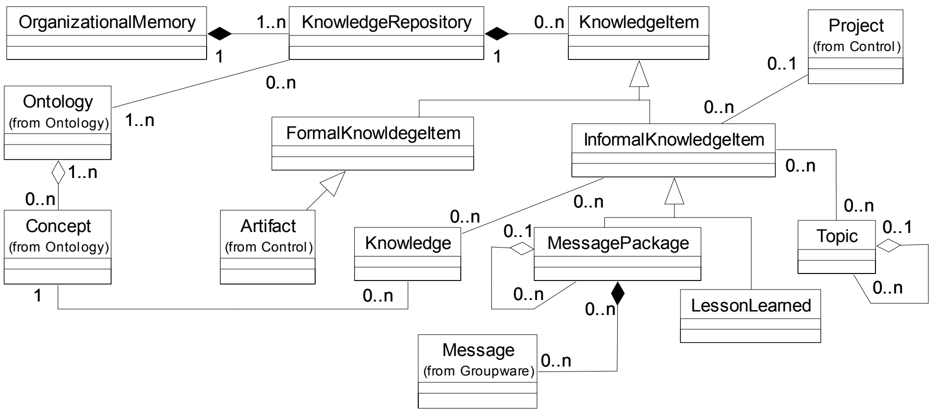


Fig. 3. ODE's Organizational Memory Structure (partial model).

ODE's knowledge items are classified into formal and informal items. Formal items are artifacts created during the software process. Informal items include lessons learned and message packages. Lessons learned are reports describing successful solutions adopted to solve problems, or improvement opportunities. Message packages store important discussions made by developers during software projects developed using ODE. Lessons learned and message packages must be evaluated and, optionally adapted, before getting into ODE's OM. Also, those items are classified according to the indexing schema of ODE's informal knowledge items that includes defining related ontology instances (Knowledge), topics and, optionally, projects, as shown in Figure 3.

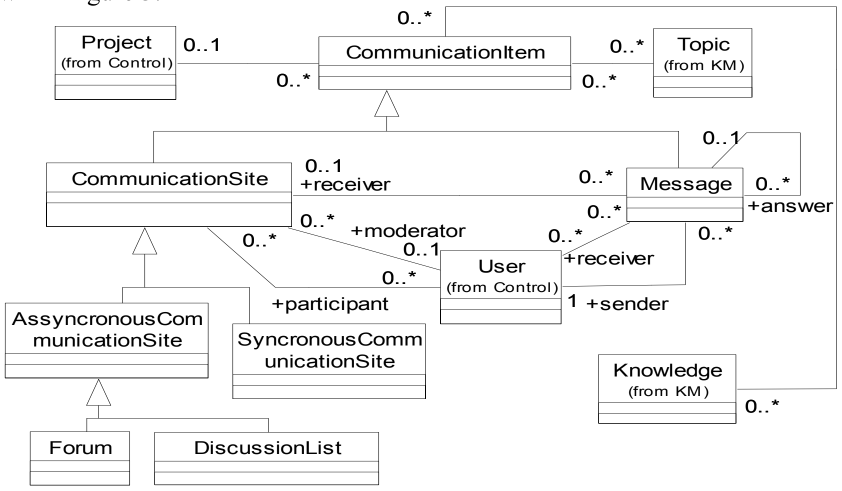
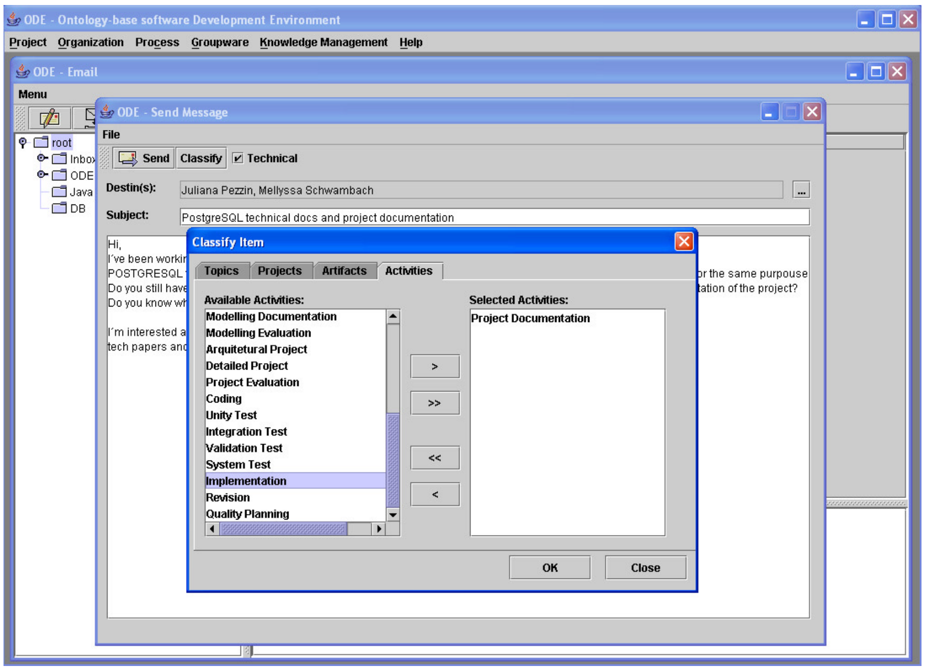


Fig. 4. ODE's Groupware Facilities.

Message packages are obtained by evaluating and adapting the messages exchanged in ODE's groupware tools. Groupware in ODE includes an instant messaging tool, an email tool with discussion lists, and a forum tool. These communication sites store the exchanged messages, and are mapped in ODE's internal model to Synchronous Communication Site, Discussion List and Forum, respectively, as shown in Figure 4. Using these tools, developers must classify their messages, in order to allow them to be packaged later. Figure 5 shows an email being classified. According to the indexing schema of ODE's informal knowledge items, first the user should select which concepts of the ontology are involved in the message. Based on these concepts, ontology instances (objects from the Knowledge class) are presented. In figure 5, two concepts were selected (Activity and Artifact). Thus instances of these Knowledge classes can be used to classify this message. Also, as shown in figure 5, by default, Topics and Projects can be used to classify messages.



Destin(s): Juliana Pezzin, Mellyssa Schwambach

Subject: PostgreSQL technical docs and project documentation

Hi,
I've been workin
POSTGRESQL
Do you still have
Do you know wh

I'm interested a
tech papers and

r the same purpouse
tation of the project?

Available Activities: Modelling Documentation, Modelling Evaluation, Arquitetural Project, Detailed Project, Project Evaluation, Coding, Unity Test, Integration Test, Validation Test, System Test, Implementation, Revision, Quality Planning

Selected Activities: Project Documentation

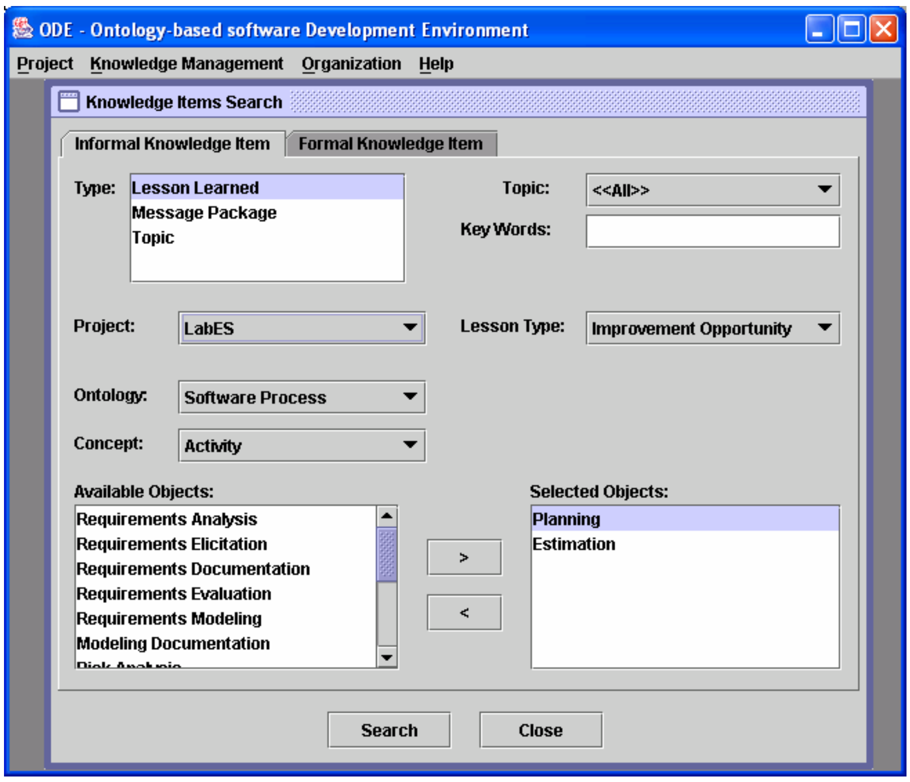
Fig. 5. Classifying messages in ODE's E-mail.

Later, classified messages can be packed by the knowledge manager, deriving packaged discussions (*MessagePackage* in figure 3), that can be searched and reused by other developers. Integrating groupware and KM, the knowledge embedded in messages, that usually is dispersed in mailboxes of stand-alone email tools, or lost in instant messaging tools, can be captured, stored, disseminated, and reused.

3.2 ODE's KM Services

As shown in Figure 2, arranged around the OM, there are services supporting the following knowledge management activities:

- **Knowledge Creation and Capture:** As discussed previously, ODE's OM contains three types of knowledge items: artifacts, lessons learned, and discussion packages. Then, ODE offers facilities to capture each one of these knowledge types. Artifacts created during the software process are submitted to configuration management and become available in ODE's central repository. Discussion packages are created by packing messages exchanged in ODE's groupware applications. This is done using the indexing schema for informal knowledge items, discussed previously. Finally, ODE's KM infrastructure offers a service to register lessons learned. As an informal item, a lesson learned must be classified, indicating project in which it was created, ontology instances associated to the lesson, and topics. Moreover, the following information must be informed: *type* (good practice, improvement opportunity or informative lesson), *context* in which the lesson occurred, description of the *problem, solution* adopted and *expected results* when reusing this item. When dealing with lessons learned, we have to consider that project-level knowledge can be useful, but it is not always the case. Generally, project-level knowledge must be handled to become an organizational knowledge. First, a developer inputs a lesson learned in the OM. At this moment, this knowledge is not available for other developers. The knowledge manager must, first, evaluate and adapt the lesson learned so that it can be considered knowledge at the organizational level. Once approved, the lesson learned is made available. Finally, instances of ODE's ontologies, although not considered knowledge items, can be captured using applications for instantiating ontologies, or ODEd [13], ODE's ontology editor that supports domain ontology building and instantiation in ODE. Topics, like ontology instances, are captured through a specific tool. In fact, they play a similar role of ontology instances in ODE, concerning indexing knowledge items.
- **Knowledge Retrieval and Access:** ODE's KM infrastructure supports knowledge items access through searching. At any time, developers can search the OM for any kind of knowledge. This search is a user-initiated search. That is, he/she has to define his/her needs (the knowledge he/she wants), these needs become a query, and knowledge items retrieved are presented. The user can browse the various knowledge items and then select and reuse one of them. Figure 6 shows this service applied to lessons learned. Since ODE's knowledge items are annotated with ontology instances (objects of the Knowledge class), the user can select them as filters for the search, as well as projects and topics, in an analogous way as classifying messages.
- **Knowledge Dissemination:** Knowledge dissemination is particularly important when users are not motivated to look for information or when they are not aware of the need for information in the first place. While knowledge search is a user-initiated search, knowledge dissemination concerns system proactive aid. I.e. the system performs a search without requiring the user to explicitly formulate a query. In this case, software agents monitor the users' actions as they work and inform them about potential relevant knowledge items. As in knowledge retrieval, users can then browse the items, and select and reuse one of them.



Type: Lesson Learned / Message Package / Topic

Topic: <<All>>

Key Words:

Project: LabES

Lesson Type: Improvement Opportunity

Ontology: Software Process

Concept: Activity

Available Objects: Requirements Analysis, Requirements Elicitation, Requirements Documentation, Requirements Evaluation, Requirements Modeling, Modeling Documentation, Risk Analysis

Selected Objects: Planning, Estimation

Fig. 6. ODE's Knowledge Retrieval Service applied for Lessons Learned.

- **Knowledge Use:** Once a search is completed and a knowledge item selected for use, the user must evaluate the item adequacy for helping knowledge maintenance. It includes some evaluation information, such as if the item was useful, problems that appeared when reusing it, and solutions applied.
- **Knowledge Maintenance:** For maintenance and evolution of the OM, it is necessary to take into account users' feedback. Based on the user feedback, the knowledge manager can decide which knowledge items are obsolete or which ones had never been used. To perform this task, the knowledge manager has an interface to search for knowledge items, and to exclude them. A software agent can be set to alert the knowledge manager to realize an OM's maintenance at defined time intervals, or when the OM has reached a defined size. The software agent can also suggest some knowledge items to be excluded based on some knowledge manager criteria.

It is worthwhile to point out that ODE's KM services are classified in two types: general services and tool-specific services. General services include those related to knowledge capture, retrieval, use and maintenance. These services are called "general services", because they are provided by the environment, and they are available from the environment as much as from all its tools. Tool-specific services, in turn, concern knowledge dissemination. These services need to take into account features and working of a specific tool, because it is not possible to offer pro-active aid without knowing details about the task being supported by the tool. Thus, knowledge dissemination must be developed for each one of the ODE's tools.

Agents are used to implement the dissemination services. Each tool has an agent (or a community of agents) that monitors developers using it. When the agent perceives that there are knowledge items that can help the user, it acts, showing them. In order to support developing those agents, a framework for building agents was developed, as shown in Figure 7.

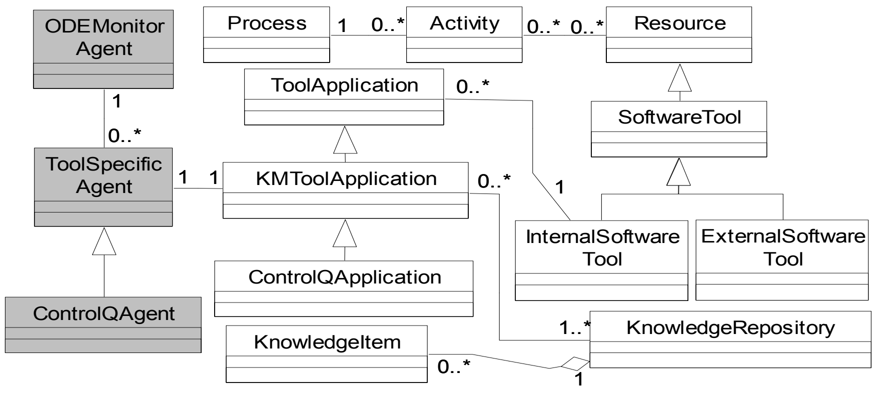


Fig. 7. ODE's Knowledge Dissemination Framework.

The dissemination service starts to be carried through the functioning of ODE's personal agent that monitories user action and verifies when a tool with KM support is initiated. From this moment, it is necessary that the tool specific agent follows user actions to know when to present knowledge and what is considered relevant knowledge for the task in hand.

Since only an agent cannot know the internal functioning of all ODE's tools, it is necessary that each tool with KM support has its own agent, defined and configured during tool development time. Therefore, in order to facilitate implementing knowledge dissemination services in ODE's tools, the framework for building dissemination agents was developed.

In ODE, each internal software tool has a main application through which it is executed. To respect the ODE's integration politics, these applications of internal software tool must belong to a hierarchy of class, whose super class is *ToolApplication*. Thus, the main application of each tool must inherit from *ToolApplication*. When an ODE's internal tool has KM support, it has to have its main application inheriting from the *KMToolApplication* class. As shown in Figure 7, the application *ControlQ*, a tool that supports software quality planning and control [6], inherits from this class. Both instances of the *KMToolApplication* class (that is, software tools main applications), and of the *ODEApplication* class (that is, the whole environment) have access to knowledge repositories.

Moreover, it is necessary to create a specialization of the *ToolSpecificAgent* class, associating it to the tool that it should support. In this specialization, it must be defined what to present (type of knowledge items) and when to present the knowledge, implementing the methods *searchKnowledge()* and *presentKnowledge()*, respectively. As shown in Figure 7, the *ControlQAgent* class implements the specific agent to support ControlQ tool. This specific agent is activated by the *ODEMonitorAgent* when ControlQ is initiated. Thus, the specific agent is able to act proactively, searching and disseminating knowledge.

4 Related Works

Several works have exploited the use of KM systems to support software engineering tasks, such as [2, 14]. Borges et al. [2] store and share the experience obtained in software process definition. To share this knowledge, an experience repository was built, containing the organizational standard process, as well as the artifacts and lessons learned obtained throughout the projects. In order to facilitate the storage and sharing of the experience, they built ProKnowHow, a tool that supports the standard software process tailoring procedure for each project, providing KM support.

In [14], a system for supporting experience management in Q-Labs, a multinational software improvement consultancy, is presented. The objective is to provide a "virtual office" for Q-Labs, and to allow Q-Labs consultant to benefit from the experience of every other Q-Labs consultants.

Looking to these works, we can find many common points. All of them, including ours, are based on the concept of Experience Factory [15], i.e. an organizational unit that supports reuse of experience and collective learning by developing, updating and providing, on request, past experiences to be used by project organizations. However, none of them is integrated to a Software Development Environment (SDE). Thus, it is worth to remember that this work was developed in the context of ODE, an ontology-based SDE. The remarkable feature of our work is proposing a KM approach actively integrated into the work process and collaboration practices of a SDE. A major goal is to capture information from the work process without extra effort for developers who can receive knowledge from an active organizational memory.

Observing structural aspects of a KM system, we also find many related works in the literature. Abecker et al. [16], for example, defined a KM approach with an organizational memory at the core of the KM system. Arranged around such an organizational memory, KM services provide actively knowledge to users. Our approach shares many of the definitions proposed by them. Thus, ODE's KM infrastructure has also the organizational memory acting as a central knowledge repository and around it, there are services for capturing, searching, disseminating, using and maintaining knowledge.

Ontologies have been pointed as crucial for KM systems. Benjamins et al. [17], for example, present a KM approach based on ontologies and use ontological engineering to organize and structure knowledge. Ontologies also play an important role in our approach, since they are used to structure ODE's organizational memory. But, in our approach, ontologies also give rise to knowledge items, since ontologies can be instantiated.

Several researches pointed out the benefits of software agents for several purposes in KM. Staab et al. [18], for example, present an approach for intelligent proactive knowledge dissemination. Agents work on knowledge created through the usual work tasks of the user and offer knowledge that may be relevant for his/her currently task.

In our approach, agents also disseminate knowledge according to users' needs. But in contrast, we embed our agent support in specific steps of activities, based on its ontological distinctions, using semantic information to guide dissemination.

Finally, concerning groupware, there are several tools that have facilities to exchange, store, classify and index messages, but generally messages are not put in an organizational memory, but they are dispersed on users' mailboxes. On the other hand, in ODE, messages are packaged and stored as knowledge items that can be searched and disseminated like any other ODE's informal knowledge item.

5 Conclusions

Knowledge management systems facilitate access and reuse of knowledge typically by using several emerging technologies, such as ontologies and software agents. In this paper we presented an infrastructure for managing knowledge in a software development environment (SDE) called ODE.

In ODE's KM approach, knowledge workers constantly create new knowledge as they work. Some benefits of this approach can be pointed out: (i) With KM integrated to a SDE, it is easier for developers to create new knowledge. In this way, the organizational memory is not closed. It is always evolving. (ii) A major concern for KM in ODE is to capture knowledge during the software process without developers' extra effort. Thus, the KM system is actively integrated into the work process. An isolated KM system, on the other hand, can be a barrier to innovation, because it does not let workers share new ideas with their peers. Closed systems do not give organizations control over their own knowledge, since there is a gap between knowledge creation and integration. Innovations happen outside the KM system, and then it contains information that is chronically out of date and that reflects an outsider's view of work [8]. (iii) Developers (as knowledge workers) are no longer passive receivers of knowledge, but are active researchers, constructors, and communicators of knowledge. Knowledge can be constructed collaboratively in the context of the work. Attention to knowledge requires attention to people, including their tasks, motivation, and interests in collaboration. The heart of intelligent human performance is not the individual human mind, but groups of minds interacting with each other and with tools and artifacts [8]. (iv) A KM system must provide the information workers need, when they need it. ODE's KM based tools play an active role in knowledge dissemination. Software agents monitor the actions of users as they work, and inform them about potentially relevant knowledge for the task at hand.

In November 2004, ODE was implanted in a software house as a pilot project, and we expect that soon we can discuss actual results derived from its use, especially those concerning the practical usefulness of its KM approach.

Acknowledgments

This work was accomplished with the support of CNPq and CAPES, entities of the Brazilian Government reverted to scientific and technological development.

References

1. D.E. O'Leary, "Enterprise Knowledge Management", IEEE Computer Magazine, March, 1998.
2. L.M.S. Borges, R.A. Falbo, "Managing Software Process Knowledge", Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'2002), pp. 227 – 232, Foz do Iguazu, Brazil, June 2002.
3. M.H. Zack, M. Serino, "Knowledge Management and Collaboration Technologies", in *Knowledge, Groupware and the Internet*, Butterworth-Heinemann, 2000, pp. 303-315.
4. D.E. O'Leary, R. Studer, "Knowledge Management: An Interdisciplinary Approach", IEEE Intelligent Systems, January/February, vol. 16, No. 1, 2001.
5. W. Harrison, H. Ossher, P. Tarr, "Software Engineering Tools and Environments: A Roadmap", in Proc. of the Future of Software Engineering, ICSE'2000, Ireland, 2000.
6. R.A. Falbo, A.C.C. Natali, P.G. Mian, G. Bertollo, F.B. Ruy. "ODE: Ontology-based software Development Environment", Proceedings of the IX Argentine Congress on Computer Science (CACIC'2003), La Plata, Argentina, 2003, pp 1124-1135.
7. A. Tywana. Knowledge Management Toolkit: Orchestrating IT, Strategy, and Knowledge Platforms, 2nd edition, Prentice Hall PTR, 2002.
8. G. Fischer, J. Ostwald, "Knowledge Management: Problems, Promises, Realities and Challenges", IEEE Intelligent Systems, vol. 16, No. 1, January/February, 2001.
9. R.A. Falbo, C.S. Menezes, A.R.C. Rocha. "A Systematic Approach for Building Ontologies". Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.
10. R.A. Falbo, G. Guizzardi, G., K.C. Duarte, "An Ontological Approach to Domain Engineering", in Proc. of the 14th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'02, Ischia, Italy, 2002.
11. R.A. Falbo, F.B. Ruy, G. Bertollo, D.F. Togneri, "Learning How to Manage Risks Using Organizational Knowledge", Advances in Learning Software Organizations (Proceedings of the 6th International Workshop on Learning Software Organizations - LSO'2004), Melnik G. and Holz, H. (Eds.): LNCS 3096, pp. 7-18, Springer-Verlag Berlin Heidelberg, Banff, Canada, June 2004.
12. S. Staab, R. Studer, H.P. Schurr, Y. Sure, *Knowledge Processes and Ontologies*, IEEE Intelligent Systems, January/February, Vol. 16, No. 1, 2001.
13. P.G. Mian, R.A. Falbo. "Building Ontologies in a Domain Oriented Software Engineering Environment". Proceedings of the IX Argentine Congress on Computer Science, La Plata, Argentina, 2003, pp. 930 – 941.
14. M.G. Mendonça Neto, V. Basili, C.B. Seaman, and Y-M Kim, "A Prototype Experience Management System for a Software Consulting Organization", in Proc. of the 13th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'01, Buenos Aires, Argentina, 2001.
15. V. Basili, G. Caldiera, H. Rombach. "The Experience Factory", Vol. 1 of Encyclopedia of Software Engineering, Chapter X, John Wiley & Sons. 1994.
16. A. Abecker, A. Bernardi, K. Hinkelman. "Toward a Technology for Organizational Memories", IEEE Intelligent Systems, Vol. 13., No. 3, pp. 40-48, 1998.
17. V.R. Benjamins, D. Fensel, A.G. Pérez, "Knowledge Management through Ontologies", Proc. of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98), Switzerland, 1998.
18. S. Staab, H. P. Schurr. "Smart Task Support through Proactive Access to Organizational Memory". Knowledge-based Systems, 13(5): 251-260. Elsevier, 2000.