...rch, development, and education, at a high level and in both printed ...electronic form. Enjoying tight cooperation with the R&D community, ...numerous individuals, as well as with prestigious organizations and ...ties, LNCS has grown into the most comprehensive computer science ...rch forum available.

...cope of LNCS, including its subseries LNAI and LNBI, spans the ... range of computer science and information technology including ...isciplinary topics in a variety of application fields. The type of ...ial published traditionally includes

...eedings (published in time for the respective conference)

...-proceedings (consisting of thoroughly revised final full papers)

...rch monographs (which may be based on outstanding PhD work, ...rch projects, technical reports, etc.)

...ecently, several color-cover sublines have been added featuring ...l a collection of papers, various added-value components; these ...s include

...ials (textbook-like monographs or collections of lectures given at ...nced courses)

...of-the-art surveys (offering complete and mediated coverage ...opic)

...pics (introducing emergent topics to the broader community)

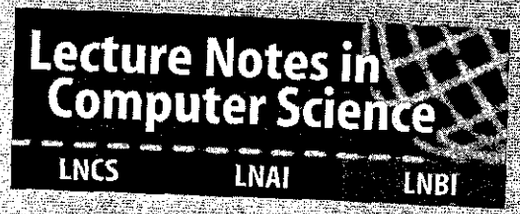
...llel to the printed book, each new volume is published ...fically in LNCS Online.

... information on LNCS can be found at ...ww.springeronline.com

...ls for publication should be sent to ...litorial, Tiergartenstr. 17, 69121 Heidelberg, Germany ...lncs@springer.de

...92-9743

...online.com



...lnik · Holz (Eds.)

LNCS 3096

Advances in Learning Software Organizations

LSO 2004

LNCS 3096

Advances in Learning Software Organizations

6th International Workshop, LSO 2004
Banff, Canada, June 2004
Proceedings

Springer

:r

Advances in Learning Software Organizations

6th International Workshop, LSO 2004
Banff, Canada, June 20-21, 2004
Proceedings

Springer

Volume Editors

Grigori Melnik
University of Calgary, Department of Computer Science
2500 University Dr. N.W., Calgary, Alberta T2N 1N4, Canada
E-mail: melnik@cpsc.ucalgary.ca

Harald Holz
German Research Center for Artificial Intelligence (DFKI) GmbH
Knowledge Management Department
Erwin-Schroedinger-Str. 57, 67663 Kaiserslautern, Germany
E-mail: harald.holz@dfki.de

Library of Congress Control Number: 2004106916

CR Subject Classification (1998): D.2, K.6, H.5.2-3, I.2.4, K.3, K.4.3

ISSN 0302-9743
ISBN 3-540-22192-1 Springer-Verlag Berlin Heidelberg New York

Preface

Software-intensive organizations cannot help but learn. A software organiz does not learn will not exist for long, because the software market is continu the move, because of new customer demands and needs, and because of new co products and services. Software organizations must adapt quickly to this ever-c environment, and the capability to adapt is one of the most important aspects ing. Smart organizations will attempt to predict future software demands, and a corresponding knowledge road map that identifies the capabilities needed o in order to meet these demands.

Organizational learning typically occurs when experienced organization r share their knowledge with colleagues, such that the organization as a whole c from the intellectual capital of its members. While knowledge is typically sl an ad hoc fashion by means of direct, face-to-face communication, a learning s organization will want to ensure that this knowledge sharing occurs in a systema enabling it whenever and wherever it is needed.

Since 1999, the annual International Workshop on Learning Software Organ (LSO) has provided a communication forum that brings together academia and i to discuss the advancements in and to address the questions of continuous lea software-intensive organizations. Building upon existing work on knowledge r ment and organizational learning, the workshop series promotes interdisciplir proaches from computer science and information systems, business, managem organization science as well as cognitive science.

The LSO workshop series differs from other conferences in that it puts a ular focus on organizational (rather than mere technological) questions such a to introduce knowledge management approaches in a software organization i mally intrusive ways? What IT support is successful in fostering a culture of kno sharing? What roles are required to effectively disseminate available knowledg can information overload be avoided, while ensuring that relevant informatior overlooked? What knowledge management approaches scale up to the needs o multisite organizations? Approaches that address these issues often have to inte number of diverse techniques, methods, and tools.

This volume contains 13 full papers and 3 short papers selected by the pı committee for presentation at the 6th International Workshop on Learning Sc Organizations (LSO 2004). The workshop was held in conjunction with the 1€ ternational Conference on Software Engineering and Knowledge Engineering (2004) which allowed for further fruitful, inter-disciplinary discussions betwee groups of participants. In addition to the oral presentation of the papers contai this volume, the LSO 2004 program included a keynote talk by Philippe Kruchte

As the workshop chairs [illegible]

the submitted papers. Last but not least, we would especially like to thank our keynote speaker Philippe Kruchten for sharing his insights with us.

We hope that this volume conveys to its readers at least some of the vital and interesting discussions that were stimulated by the presentation given at the LSO 2004 workshop.

June 2004

Grigori Melnik
Harald Holz

Workshop Organization

Workshop Chairs

Grigori Melnik, University of Calgary, Calgary (Canada)
Harald Holz, DFKI GmbH, Kaiserslautern (Germany)
Scott Henninger, University of Nebraska-Lincoln, Lincoln (USA)

Program Committee

Klaus-Dieter Althoff, Fraunhofer IESE (Germany)
Ralph Bergmann, University of Hildesheim (Germany)
Andreas Birk, sd&m AG (Germany)
Raimund L. Feldmann, Fraunhofer Center Maryland (USA)
Christiane Gresse von Wangenheim, Universidade do Vale do Itajaí (Brazil)
John C. Grundy, University of Auckland (New Zealand)
Frank Houdek, DaimlerChrysler AG (Germany)
Franz Lehner, University of Regensburg (Germany)
Mikael Lindvall, Fraunhofer Center Maryland (USA)
Makoto Matsushita, Osaka University (Japan)
Frank Maurer, University of Calgary (Canada)
Günther Ruhe, University of Calgary (Canada)
Ioana Rus, Fraunhofer Center Maryland (USA)
Martin Schaaf, University of Hildesheim (Germany)
Markus Strohmaier, The Know-Center, Graz (Austria)
Carsten Tautz, empolis GmbH (Germany)
Rosina Weber, Drexel University (USA)
Yunwen Ye, University of Colorado at Boulder (USA)

Additional Reviewers

Michael M. Richter, University of Kaiserslautern (Germany)
Torsten Willrich, Fraunhofer IESE (Germany)

Steering Committee

Klaus-Dieter Althoff, Fraunhofer IESE (Germany)
Raimund L. Feldmann, Fraunhofer Center Maryland (USA)
Scott Henninger, University of Nebraska-Lincoln (USA)

Sponsoring Institutions

The organizers of LSO 2004 would like to thank the following institutions for their generous support:

- informatics Circle of Research Excellence (iCORE), Canada
- University of Calgary, Canada
- German Research Center for Artificial Intelligence (DFKI) GmbH, Germany

Furthermore, the organizers thank for administrative support and logistics:

- Camille Sinanan, University of Calgary (Canada)
- The Banff Centre (Canada)

LSO Workshop Series

Materials of past and upcoming workshops in the Learning Software Organizations series are available from:

http://www.iese.fhg.de/Publications/lso

Table of Contents

Introduction

Experience-Based Information Systems

Software Maintenance

Communities of Practice

Research on Learning Software Organizations - Past, Present, and Future

Harald Holz[1] and Grigori Melnik[2]

[1] DFKI GmbH, Knowledge Management Department,
PO Box 2080, 67608 Kaiserslautern, Germany
Harald.Holz@dfki.de
[2] Department of Computer Science, University of Calgary,
Calgary, Canada
melnik@cpsc.ucalgary.ca

Knowledge itself is power, not mere argument or o
Francis Bacon (Meditations Sacra

1 Introduction

In order for a software organization to stay competitive, its software deve needs to be part of organizational change. The organization's ability to chang adapt quickly to environmental changes provides a foundation for growth an [7]. For such changes to happen, the learning capabilities of the organization be enhanced, being an essential part of producing more effective and efficie practices. Moreover, continuous learning is essential for surviving – let alone ing – in dynamic and competitive environments [15]. The Learning Software zation (LSO) workshop series has been promoting this vision since 1999, ad the questions of organizational learning from the software development point c

Though the workshop series is relatively young, the ideas it is based on ha circulating for decades. As early as in 1971, Weinberg recognized software ment as learning: "writing a program is a process of learning – both for the p mer and the person who commissions the program" [23]. This was supersede engineering approach, when software development began to be considered ware engineering", omitting for a long period the humanistic people-centric a it. The history of LSO workshops reflects this development to a certain de 1999, LSO started with the premise that "with continuous technological chan alization, business reorganizations, e-migration, etc. there is a continuous sh the right knowledge at the right place at the right time. To overcome this sh the challenge for the Learning Organization." [20]. In other words, the main c considered six years ago was the *availability* of knowledge. As a result, ma tions were built to address this. The proliferation of knowledge bases is a clea tion of this. The knowledge is primarily considered to be an object, and, thu be codified, stored, retrieved and distributed. Unfortunately, many such suffered from the "build it and they'll come" syndrome which resulted in a user involvement and enthusiasm. Researchers and practitioners in LSO

How to Manage Knowledge in the Software Maintenance Process

Oscar M. Rodríguez[1], Aurora Vizcaíno[2], Ana I. Martínez[1], Mario Piattini[2], and Jesús Favela[1]

[1] CICESE, Computer Science Department, México
{orodrigu,martinea,favela}@cicese.mx
[2] Alarcos Research Group. University of Castilla-La Mancha, Escuela Superior de Informática, España
{Aurora.Vizcaino,Mario.Piattini}@uclm.es

Abstract. The software maintenance process involves a lot of effort and costs. In fact, this stage is considered the most expensive of the software development life-cycle. Moreover, during maintenance a considerable amount of information needs to be managed. This information often comes from diverse and distributed sources such as the products to be maintained, the people who work in this process, and the activities performed to update the software. However, very few software companies use knowledge management techniques to efficiently manage this information. Appropriate knowledge management would help software companies improve performance, control costs and decrease effort by taking advantage of previous solutions that could be reused to avoid repeating previous mistakes. This work presents a multiagent system designed to manage the information and knowledge generated during the software maintenance process; using web technologies to support this management. The system has different types of agents, each devoted to a particular type of information. Agents use different reasoning techniques to generate new knowledge from previous information and to learn from their own experience. Thereby the agents become experts in the type of knowledge they are responsible for. Additionally, agents communicate with each other to share information and knowledge.

Keywords: Knowledge management, software maintenance, agents

1 Introduction

Knowledge is fast becoming the key to survival and competitive advantage [12]. Many innovative companies have long appreciated the value of knowledge to enhance their products and customer services. Therefore, a huge investment is being done in the field of knowledge management, for example, Berztiss in [5] claims that by the year 2004 the cost of knowledge management is expected to reach USD 10,200,000,000.

The software organizations, encouraged by the idea of improving costs, schedules and quality of their products, and improved customer satisfaction, are also interested in knowledge management [22]. Some reasons of this interest are that:

G. Melnik and H. Holz (Eds.): LSO 2004, LNCS 3096, pp. 78–87, 2004

a) Software Engineering is a knowledge-intensive work where the main c what has been called the "intellectual capital". Unfortunately, the owner intellectual capital are often the employees instead of being the compan might expect. Employees, from their experience, obtain tacit knowledge, richer and more valuable than explicit knowledge, but that cannot be e pressed or communicated [13]. Thereby, software organizations depend g knowledgeable employees because they are the key to a project's success.

b) Software development is a constantly changing process. Many people different phases, activities and projects. Knowledge in software engineeri verse and its proportions immense and steadily growing [15]. Organizatio have problems identifying the resources, localizations and use of knowledg

Both reasons are also applicable to a specific process of the software life cycl software maintenance where the problems mentioned above could be even m nificant. Maintainers have to face legacy software, written by people from oth which often has little or no documentation describing the features of t ware [24].

Thus, a well-known issue that complicates the maintenance process is th and distributed documentation that exists related to a specific software sy even if detailed documentation was produced when the original system wa oped, it is seldom updated as the system evolves.

Storing knowledge helps to reduce these problems since it decreases dep on employees' cognition because at least some of their expert knowledge h retained or made explicit. Moreover, storing good solutions to problems or learned avoids repeating mistakes and increases productivity and the likelih further success [22]. However, for information to be usable it needs to be m structured, generalised and stored in a reusable form, to allow for effec trieval [1].

This work describes how to manage distributed knowledge and information ated during the software maintenance process in order to improve maintaine and efficiency. The content of this paper is organized as follows: Section 2 the tasks performed during the software maintenance process and justifi knowledge management should be used in this process. Section 3 presents a agent system designed to encourage and facilitate the reuse of previous experi software maintenance organizations, using a global web repository. Finally, sions and future work are presented in Section 4.

2 Software Maintenance

Many studies [17,19] have demonstrated that most of the overall expenses i during the life-cycle of a software product occur during the maintenance p During this process different types of maintenance could be required: correcti ceptive, adaptive or preventive. Each type of maintenance has its own features of them follow a similar process, summarized in Figure 1; the maintenance e receives the requirements that the modification should fulfil. Then, s/he id which parts of the system should be modified, which modules could be affe this modification and plan what activities have to be performed. The maintain

consciously, takes advantage of his/her experience to carry out all of these tasks. And, in the case of his/her experience not being enough, the engineer would consult other resources that are often two: a person who has already solved a similar problem or has worked with that software previously or the engineer analyses the source code which means to dedicate a lot of time to this activity.

To carried out the present work, two case studies were performed [20]. In these studies, maintenance engineers were observed performing their work. The study helped us to identify scenarios that show that on many occasions, organizations had documents or people with the information or knowledge necessary to support or help other colleagues in their activities, but either the former did not know what the latter was working on, or the latter did not know that other documents or people could have provided useful information to help them to complete the assignment.

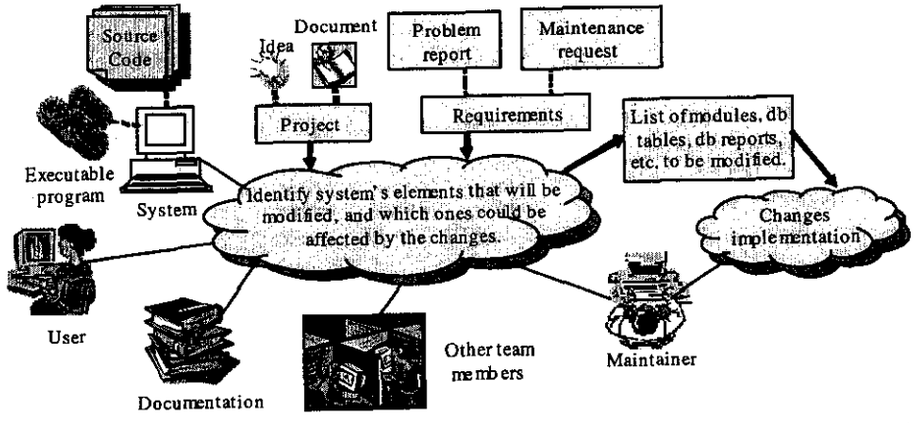


Fig. 1. Basic aspects of the process that the maintenance engineer performs at the moment of implementing changes on the system, as well as the knowledge sources that help him to do these tasks.

This fact has already been commented on by other authors, such as Szulanski [23] who found that the number one barrier to knowledge sharing was "ignorance": the sub-units are ignorant of the knowledge that exists in the organization, or the sub-units possessing the knowledge are ignorant of the fact that another sub-unit needs such knowledge. Sometimes the organization itself is not aware of the location of the pockets of knowledge or expertise [14].

After studying the results of the case studies, the question of how to help the maintainers to identify knowledge sources that could help them to carry out their work, or to improve it by decreasing costs, time or effort, arose. The scenarios identified in these studies also showed us how a knowledge management system can support some of the necessities that maintainers have while they perform their job. The analysis of these scenarios and the findings of the studies provided some basic characteristics that a knowledge management system to support software maintenance teams should fulfill. These characteristics were used to define a multi-agent architecture which is described in the next section.

3 A Multi-agent System to Manage Knowledge during the Software Maintenance Process

A knowledge management "program" in an organization often consis a strategy, processes and tools. Next we describe how we tackle these order to obtain a suitable knowledge management approach.

The knowledge management strategy in organizations can be d their goals, and how they proceed to achieve them. Software develo tions are concerned with controlling costs, meeting deadlines or/and ity. To achieve this they look for means to facilitate the work of the neers [7]. In our case, we are interested in a strategy based on m maintenance engineers to find knowledge and information that coul work. Moreover, we think that by reusing proven good solutions or engineers will increase their expertise and their work will have more costs and effort.

The second part consists of *processes* or organizational activi knowledge management. These will usually be methods for collecting knowledge. We consider that the best option to tackle this aspect is tc section of the organization in charge of these processes such as an Ex (a term introduced by Basili et al in [2]). Otherwise, we would have problems such as how to motivate maintenance engineers to capture into the system and manage it and, of course, how to reward this work.

Finally, there are many *tools* to support knowledge management an sifications of them. A generic classification to information retrieval could be also applied to knowledge management tools consist in div "active" or "passive" [8]. The first are those that notify users when they will require some kind of knowledge without their request. Pass a user to actively seek knowledge without any system support.

We consider active tools more appropriated for the software main since, as was previously mentioned, maintainers seldom know all the the organization has, and for this reason, they do not know what the search for. Therefore, the system should automatically show informa useful for a maintainer who is working on a specific project.

On the other hand, thanks to the huge use of The World Wide W for the dissemination of information, a new branch of Software E arisen, named Web Engineering, concerned with establishing sound niques, and tools for the development and maintenance of systems, plications over the Web.

As the web becomes the preferred medium for the deployment of s tions CASE tools are migrating to the web and several others have The use of the web as a platform for the support of software develop eral advantages:

- Ubiquitous access. Web repositories can be accessed from any com to the Internet, including portable devices.
- Simple integration of tools. The web is based on simple, open proto easily integrated in CASE tools to enable their interoperability. At of integration a tool can export content to HTML or semantical documents.

- Support for distributed software development. Large-scale software systems are increasingly being developed by distributed teams of specialists that need to communicate and coordinate their activities. The web offers a simple middleware for the deployment of workflow and groupware tools that support collaboration.

Because of this, we chose web servers as repositories for products and processes information. The centralized storage of this information facilitates configuration management, quality control, project tracking, and the management of maintenance requirements.

Applications manipulating Web data require both, documents or information retrieved from the Web, and metadata about this information. In order to specify and manage meta-data we used the standard MOF proposed by the OMG [16] and described in [24].

Based on the aspects just discussed and the findings of the case study presented in [20], we have identified the following requirements for our system:

- The tool should be active.
- The tool should support access to different and distributed sources of knowledge.
- The tool should support the search for solutions to similar problems and lessons learned from distributed locations (web repository).
- The tool should support the identification of modules or files which could be affected by the changes performed.
- The tool should enable the integration with other CASE tools and other software maintenance repositories.

3.1 Architecture of the System

In order to design the multi-agent architecture the MESSAGE (Methodology for Engineering Systems of Software agents) [6] methodology was used. It proposes different level of analysis. At level 1 analysis focuses on the system itself, identifying the types of agents and roles, which are described in the next paragraphs. The architecture has five main types of agents (see Figure 2): staff, product, client, project and directory agents.

The *staff agent* handles information related to the activities that the maintenance engineer (ME) performs. Its main role is to provide support to the ME in the accomplishment of his job. It monitors the ME activities, and based on this, requests the KMA to search for knowledge sources that can help the ME to perform his job. The staff agent collaborates with the KMA and KSMA that are located in its same container, and with the product and project agents that are in charge of the products and projects to which the ME is assigned.

The *product agent*, as the name suggests, manages information related with a product, including its maintenance requests, the staff members that are assigned to the product, the roles that they play, and the main elements that integrate the products (documentation, source code, databases, etc.). There is one product agent per product to be maintained. The main role of this agent is to monitor the activities that the staff members perform on each element that integrate the product. For example, which elements they consult or modify, which maintenance requests or error reports they manage, etc. In this way, the product agent has knowledge that can be used to identify, for instance, who is modifying or consulting some element of the product at each

moment. This information can be used to infer which staff membe element of the product, also the level of knowledge they have a element. To accomplish its job, this agent collaborates with severa as: the KMA and KSMA (located in its same container), the proje charge of the maintenance requests or error reports related to the p agents of the MEs assigned to the product.

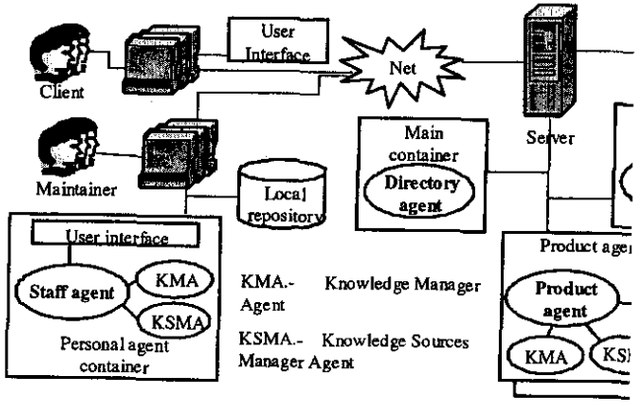


Fig. 2. Agent based architecture for a software maintenance knowledge

There is one *project agent* per project. The project agent has in tasks that must be performed to attend a maintenance request or e that these tasks could consume, the staff members assigned to the task, etc. The main role of this agent is to monitor the progress of ing to each staff member the tasks that they must perform. To c agent collaborates mainly with the staff agents of the MEs assigne

The *client agent* manages information related to the maintenan reports performed by one client. There is one agent of this kind role is to help the client when he sends an error report or a main recting it to the corresponding product agent. Another important a is to inform to the client of the state of the maintenance requests previously by him. To do these activities, this agent collaborat product and project agents.

The *directory agent* manages information required by agent communicate with other agents that are active in the system. Th type, name, and direction of all active agents. Its main role is to agents that are active in the system at each moment, to do this, must communicate with the staff, client, product, and project agent

Two auxiliary types of agents are considered in the architect *Manager Agent* (KMA) and the *Knowledge Source Manager Agen*

The KMA is in charge of providing support in the generation o search of knowledge sources. These kinds of agents are in char knowledge base in the global web repository. The staff KMA ge edge from the information obtained from the MEs in their daily w a ME is modifying a program developed in the Java language, the

the ME has knowledge about this language, and, add his/her name to the knowledge base as a possible source of knowledge about Java. On the other hand, the product KMA generates knowledge related to the activities performed on the product. It could identify patterns on the modifications done to the different modules, for example, it could identifies if there are modules or documents that are modified or consulted when a specific module is modified, and in this way, it could detect which modules or programs can be affected by the changes done on others, and which documents are related with these last ones.

Finally, the KSMA has control over the knowledge sources, such as documents. It knows the physical location of those sources, as well as the mechanisms used to consult them. Its main role is to control the access to the sources. The documents located in the workspace of the MEs, or those that are part of a product, such as the system or user documentation, are accessed through this agent. The KSMA is also in charge of the recovery of documents located in places different from its workspace. If those documents are managed by another KSMA, the first KSMA should communicate with the last one to request the documents. These kinds of agents mainly collaborate with other KSMA, and with the staff or the product agents that are in its same container.

3.2 Agents Collaboration

As we mentioned before, agents must collaborate with others in order to complete their jobs. In this section we present two examples of how this occurs. The first example shows how the ME is informed about the tasks that he must perform in a project when he starts to work on it. As Figure 3 shows, when the staff agent requests to be registered in the directory, the directory agent identifies in which product (or products) is the ME working. Then, the directory agent informs to the product agent that the ME has been registered in the system. After that, the product agent identifies the projects to which the ME is assigned, and informs to the corresponding project agents that the ME is working on the system. Finally, the project agent informs to the staff agent the tasks that must be performed by the ME.

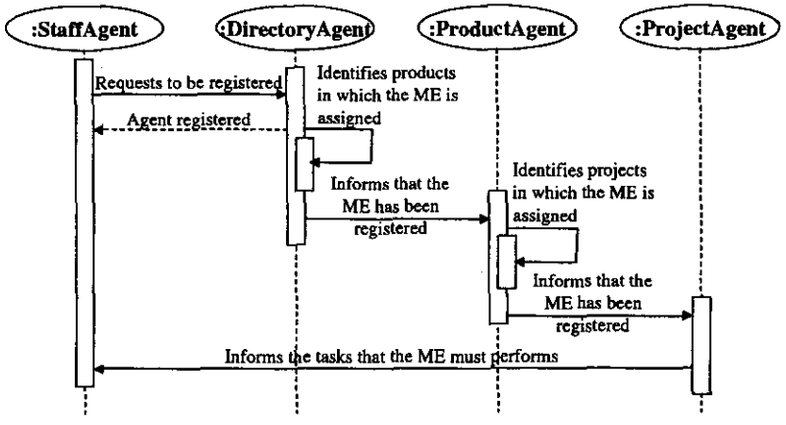


Fig. 3. Sequence diagram indicating how agents communicate to inform to the maintenance engineer the tasks he must perform.

The second example shows how the staff agent and the KMA co tween them to support the ME in the search of knowledge sources th As Figure 4 shows, the staff agent captures each event that is trigge the graphical user interface (GUI). When the staff agent identifies working on a specific task of a project, it obtains information abou example, the type of project (maintenance request or error report), module to be modified or in which the error was presented, etc. Later asks the KMA to search for knowledge sources that know about that p kind of modification or error, the language in which the product or veloped, etc. The KMA searches the knowledge base for knowledge have information related to the topics of the task to be performed. finds the sources, it informs the staff agent about them. Next, the staf the ME of the relevant sources that were found.

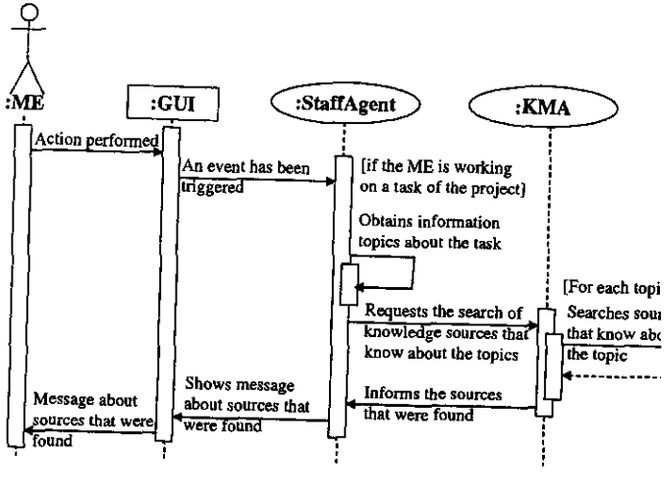


Fig. 4. The staff agent and the KMA communicate each other to help the ME se sources relevant to the problem at hand.

3.3 Some Aspects of Implementation

The platform chosen to implement the multiagent system is JADE [FIPA compliant agent platform, implemented in Java and develope source project. JADE has been used in the development of other syste main of knowledge management [4, 9, 11, 18].

This platform provides a Java API that simplifies the development run in the environment of the platform. A JADE platform could be co several agents containers, each one of which could be located in a diffe facilitates the development of distributed multiagent systems.

One of JADE's main characteristics is the support of ontologies an guages. JADE provides a mechanism to define ontologies and content l manner that makes possible to convert, between ontology's elements a language representation, in a transparent way for developers. The above definition of the language with which agents will communicate, and ena handling of complex interactions between them.

On the other hand, as Figure 2 shows, the tool has two types of repositories of information. One is where local information related to specific tasks is stored and the other is a global web repository where more generic knowledge is stored. The data are classified following an ontology for software maintenance proposed by Ruiz et al in [21], which is an extension of that of Kitchenham [10].

4 Conclusions and Future Work

Knowledge is a crucial resource for organizations. It allows companies to fulfil their mission and to become more competitive. The management of knowledge and how it can be applied to software development and maintenance has received little attention from the software engineering research community so far. However, software organizations generate a huge amount of distributed knowledge that should be stored and processed. In this way, they would obtain more benefits from it. This paper presents the architecture of a multi-agent system in charge of storing and managing information, expertise and lessons learned which are generated during the software maintenance process. The system facilitates the reuse of good solutions and the sharing of lessons learned. Thereby, the costs in time and effort should decrease.

A first prototype of the system was developed based on the requirements obtained from two case studies carried out in two software maintenance groups. As future work we are planning to perform another case study in order to evaluate where the tool makes the work of maintainers easier and to what degree costs and effort are decreased.

Acknowledgements

This work is partially supported by the TAMANSI project (grant number PBC-02-001) financed by the Consejería de Ciencia y Tecnología of the Junta de Comunidades de Castilla-La Mancha, and by CONACYT under grant C01-40799 and the scholarship 164739 provided to the first author.

References

1. Althoff, K-D., Birk, A., and Tautz, C. (1997). The Experience Factory Approach: Realizing Learning from Experience in Software Development Organizations. In proceedings of the 10th German Workshop on Machine Learning (FGML 1997), University of Karlsruhe, 6-8.
2. Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The Experience Factory. In Encyclopedia of Software Engineering, Marciniak, J.J., and Wiley, J., (Eds.) pp 469-476.
3. Bellifemine, A., Poggi, G., and Rimassa, G. (2001). Developing multi agent systems with a FIPA-compliant agent framework. Software Practise & Experience, (2001) 31: 103-128.
4. Bergenti, Federico; Poggi, Agostino and Rimassa, Giovanni. (2000). Agent Architectures and Interaction Protocols for Corporate Memory Management Systems. Proceedings of the 14th European Conference on Artificial Intelligence, Workshop on Knowledge Management and Organizational Memories. pp. 39-47.
5. Berztiss, A. T. Capability Maturity for Knowledge Management (2002) Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02), pp 162-166.
6. Caire, G., Coulier, W., Garijo, F., Gómez, J., Pavón, J., Leal, F., Chai Stark, J., Evans, R., Massonet, P. (2001). Agent Oriented Analysis UML in Agent Oriented Software Engineering, pp. 119-135.
7. Dingsoyr, T., and Conradi, R. (2002). A Survey of Case Studies of th Management in Software Engineering. International Journal of Softw Knowledge Engineering. Vol. 12, No 4, 391-414.
8. Dingsoyr, T., and Royrvik, E. (2003). An Empirical Study of an Infor pository in a Medium-Sized Software Consulting Company. In Proce ternational Conference on Software Engineering (ICSE'2003), pp 84-9
9. Gandon, Fabien. (2002). A Multi-Agent Architecture For Distribute ries. Proceedings of the Sixteenth European Meeting on Cybernetic search.
10. Kitchenham, B.A., Travassos, G.H., Mayrhauser, A., Niessink, F., S Singer, J., Takada, S., Vehvilainen, R. and Yang, H. (1999). Towa Software Maintenance. Journal of Software Maintenance: Research a 365-389.
11. Knowledge On Demand (KOD), IST Project, IST-1999-12503, http://kod.iti.gr/, http://www.kodweb.org.
12. Macintosh, A. (1997). Position paper on Knowledge Asset Management http://www.ntgi.net/ntgi/y2k/kmfr.html
13. Meeham, B., and Richardson, I. (2003). Identification of Software F Management. Software Process Improvement and Practice, pp 45-55.
14. Nebus, J. (2001). Framing the Knowledge Search Problem: Whom D Why Do We Contact Them? *Academy of Management Best Papers P* h7.
15. Oliveira, K.M, Anquetil, N., Dias; M.G, Ramal, M., Meneses, R. (20 Software Maintenance. Fifteenth International Conference on Softwar Knowledge Engineering (SEKE'03), San Francisco, 1-3 July, pp 61-68.
16. OMG Meta Object Facility (MOF) Specification, v. 1.3 RT http://www.omg.org.
17. Pigoski, T.M. (1997): Practical Software Maintenance. Best Practices f Investment. Ed. John Wiley & Sons, USA.
18. Poggi, Agostino; Rimassa, Giovanni and Turci, Paola. (2002). An Int Agent System for Corporate Memory Management. Proceedings of th pean Meeting on Cybernetics and Systems Research.
19. Polo, M., Piattini, M., and Ruiz, F. (2002):Using a Qualitative Research ing A Software Maintenance Methodology. In Software Practice & Wiley and Sons. Vol. 32, N° 13, 1239-1260.
20. Rodriguez, O. M., Martinez, A. I., Favela, J, and Vizcaino, A. (2003). Conocimiento como soporte al Mantenimiento de Software. Avances Computación. ENC 2003. Tlaxcala, México, 8-12 September. pp 367-372
21. Ruiz, F., Vizcaíno, A., Piattini, M. and García, F. (2003). An Ontology ment of Software Maintenance Projects. Sent to the *International Journa gineering and Knowledge Engineering*.
22. Rus, I., and Lindvall, M., (2002). Knowledge Management in Software E Software, May/June, pp 26-38.
23. Szulanski, G., (1994). Intra-Firm Transfer of Best Practices Project. *Am ity and Quality Centre*, Houston, Texas, pp 2-19.
24. Vizcaino, A., Favela, J., Piattini, M., García, F. (2003). Supporting Softw in Web Repositories through a Multi-Agent System. In Menasalvas, E., Szczepaniak, P. S. (Eds.) First International Atlantic Web Intellige (AWIC'2003). LNAI 2663, pp 307-317.