Applying Agents to Knowledge Management in Software Maintenance Organizations

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

Abstract. This work presents a multi-agent system designed to manage the information and knowledge generated during the software maintenance process. 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. Thus, there is reuse of knowledge and knowledge management in the multi-agent architecture itself.
Keywords: Agents, Knowledge Management, Software Maintenance.

1. INTRODUCTION

Knowledge is fast becoming the key to survival and competitive advantage [13]. 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 industry, encouraged by the idea of improving costs, schedules and quality of their products is also betting on knowledge management [12]. Some reasons of this interest in knowledge management are that:

a) Software Engineering is a knowledge-intensive work where the main capital is what has been called the "intellectual capital". Unfortunately, the owners of this intellectual capital are often the employees instead of being the company as we might expect. Employees, from their experience, obtain tacit knowledge, which is richer and more valuable than explicit knowledge, but that cannot be easily expressed or communicated [14]. Thereby, software organizations depend greatly on knowledgeable employees because they are the key to a project's success. When a person with significant knowledge leaves an organization, it creates severe knowledge gaps but, what is worse, nobody in the organization knows what knowledge they have lost. "Knowing what employees know is necessary for organizations to create a

[1] CICESE, Computer Scicience Deparment, México
{orodrigu|martinea|favela}@cicese.mx

[2] Alarcos Research Group. University of Castilla-La Mancha, Escuela Superior de Informática, España

strategy for preventing valuable knowledge from disappearing" [12].

b) Software development is a constantly changing process. Many people work in different phases, activities and projects. Knowledge in software engineering is diverse and its proportions immense and steadily growing [17].

Both reasons are also applicable to a specific process of the software life cycle: this is software maintenance where the problems mentioned above could be even more significant. Maintainers have to face legacy software, written by people from other units, which often has little or no documentation describing the features of the software [25].

Thus, a well-known issue that complicates the maintenance process is the scarce documentation that exists related to a specific software system, or even if detailed documentation was produced when the original system was developed, it is seldom updated as the system evolves.

Storing knowledge helps to reduce these problems since it decreases dependency on employees' knowledge because at least some of their expert knowledge has been retained or made explicit. Moreover, storing good solutions to problems or lessons learned avoids repeating mistakes and increases productivity and the likelihood of further success [12].

This work describes how a multi agent knowledge management system may be used during the software maintenance process in order to improve maintainers' work and efficiency. The contents of this paper are organized as follows: Section 2 outlines the tasks performed during the software maintenance process and justifies why knowledge management should be used in this process. Section 3 explains why agents are a suitable technique to manage knowledge and describe a multiagent system designed to encourage and facilitate the reuse of previous experience in software maintenance organizations. Finally, conclusions and future work are presented in Section 4.

2. SOFTWARE MAINTENANCE

Many studies [18, 20] have demonstrated that most of the overall expenses incurred during the life-cycle of a software product occur during the maintenance process. During this process different types of maintenance could be required: corrective, perceptive, adaptive or preventive. Each type of maintenance has its own features but all of them follow the same process, summarized in Figure 1; the maintenance engineer receives the requirements that the modification should fulfill. Then, s/he identifies which parts of the system should be modified, which modules could be affected by this modification and plans what activities have to be performed. The maintainer, unconsciously, 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 maintainer 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 s/he analyses the source code, which means to dedicate a lot of time to this activity.

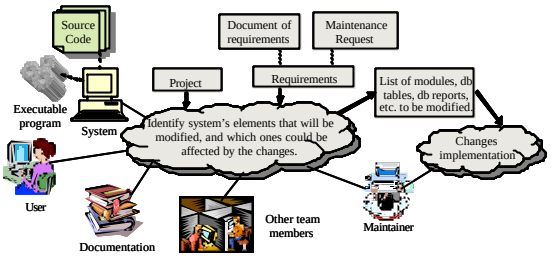

Figure 1. Summary of maintainers work

Frequently, information sources are often not consulted because people ignore their existence or location. Moreover, sometimes the organization itself is not aware of the location of the pockets of knowledge or expertise [16]. This is the number one barrier to knowledge sharing [23]. We observed this problem in two case studies carried out in two software maintenance teams. The study showed that on many occasions, organizations have documents or people with the information or knowledge necessary to support or help the maintainers to do their activities, but either the latter did not know that other documents or people could have provided useful information to help them to complete the assignment or the people with useful information did not know what the latter was working on [21].

After analyzing the results of the case study, 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.

3. A MULTI-AGENT SYSTEM TO MANAGE KNOWLEDGE DURING THE SOFTWARE MAINTENANCE PROCESS

A knowledge management "program" in a company often consists of three parts: a strategy, processes and tools. Next we describe how we tackle these three aspects in order to obtain a suitable knowledge management approach.

The knowledge management strategy in organizations can be defined based on their goals, and how they proceed to achieve them. Software development organizations are concerned with controlling costs, meeting deadlines or/and improving quality. To achieve this they look for means to facilitate the work of their software engineers [7]. In our case, we are interested in a strategy based on making easier for maintenance engineers to find knowledge and information that could facilitate their work. Moreover, we think that by reusing proven good solutions or lessons learned, engineers will increase their expertise and their work will have more quality with less costs and effort.

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

Finally, there are many *tools* to support knowledge management and different classifications of them [8]. A generic classification is to divide the tools into "active" or "passive". The first are those that notify users when it is likely that they will require some kind of knowledge without their request. Passive tools require a user to actively seek knowledge without any system support.

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

Based on the aspects just discussed and the findings of the case studies carried out, we have identified the following requirements for our system:

- The tool should be active.
- The tool should support access to different sources of knowledge.
- The tool should support the search for solutions to similar problems and lessons learned.
- The tool should support the identification of modules or files which could be affected by the changes performed.

3.1. Why Agents?

To address the need for an active knowledge management system able to generate and identify appropriate knowledge and knowledge sources, we have based our proposal on the use of agents. Agents are proactive and autonomous. This means that they can decide to act when they find it necessary to do so. Thanks to these features agents can advise users when and how should look for information in the knowledge management system. Thus, agents solve two problems about which users of this kind of systems often complain: What information should be introduced in the tool and how and when it should be consulted [12].

Moreover, agents can manage both distributed and local information. This is an important feature since in most organizations in general and in software maintenance in particular, information is generated by different sources and often from different places.

The third reason is that agents can utilize different reasoning techniques depending on the situation. For instance, by using Case Based Reasoning (CBR) the problem of finding similar information or related problems is simplified. And with the use of techniques such as induction, agents can learn which modules or files are affected by a modification.

The fourth reason for using agents is that during the maintenance process many types of knowledge should be managed such as information about the products to be maintained, about the staff working on the different projects, about the projects and so on. Agents with different roles can be in charge of each type of information and they may also share information and communicate with each other, taking advantage of the expertise of each agent.

Furthermore, agents have already been successfully used in other systems in charge of knowledge management [15, 24]. Therefore, their efficiency in these tasks has been proved.

3.2 Architecture of the System

In order to design a multi-agent system it is advisable to use a methodology that supports the development process. We have used MESSAGE (Methodology for Engineering Systems of Software agents) which combines several important features of existing agent oriented software engineering methodologies [6]. MESSAGE 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 is a mediator between the maintainer and the system. It acts like an assistant to the maintenance engineer (ME). The rest of the agents of the system communicate with the ME through this agent. The staff agent monitors the ME activities and requests the KMA to search for knowledge sources that can help the ME to perform his/her job. This agent has information that could be used to identify the ME profile, such as which kinds of knowledge or expertise s/he has or which kinds of sources s/he often consults.

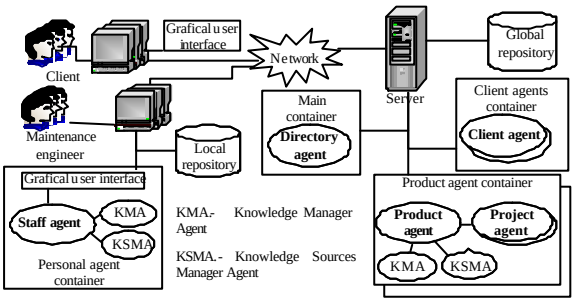

Figure 2. Agent based architecture for a software maintenance knowledge management tool.

The *product agent* manages information related to a product, including its maintenance requests and the main elements that integrate the product (documentation, source code, databases, etc.). The main role of this agent is to have updated information about the modifications carried out in a product and the people that were involved in it.

When the product agent receives a maintenance request sent by a client, it creates a new project and proposes the tasks that must be done in order to fulfill the request. The agent also proposes the most suitable people to perform those tasks and sends the proposal to the staff agent in charge to assist the ME that plays the role of project manager. The staff agent informs the ME of these proposals, and s/he decides if the proposal is accepted or modified. Once the proposal has been accepted, the project agent starts to work.

Each project is managed by a *project agent*, which is in charge of informing the MEs involved in a project about the tasks that they should perform. To do this, the project agents communicate with the staff agents. The project agents also control the evolution of the projects.

The *client agent* manages information related to the maintenance requests or problem reports performed by a client. There is one agent of this kind per client. Its main role is to assist them when they send a maintenance request, directing it to the corresponding product agent. Another important activity of this agent is to inform the client about the state of the maintenance requests sent previously by him/her, by consulting the project agents in charge of this request.

The *directory agent* manages information required by agents to know how to communicate with other agents that are active in the system. This agent knows the type, name, and electronic address of all active agents. Its main role is to control the different agents that are active in the system at each moment.

Two auxiliary types of agents are considered in the architecture, the *Knowledge Manager Agent* (KMA) and the *Knowledge Source Manager Agent* (KSMA).

The KMA is in charge of providing support in the generation of knowledge and the search of knowledge sources. This kind of agent is in charge of managing the knowledge base. The staff KMA generates new knowledge from the information obtained from the MEs in their daily work. For example, if a ME is modifying a program developed in the Java language, the KMA can infer that the ME has knowledge of 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 detect that there are modules or documents that should be modified or consulted when a specific module is modified, and in this way, it could indicate which modules or programs can be affected by the changes done on others.

Finally, the KSMA has control over the knowledge sources, such as electronic documents. It knows the physical location of those sources, as well

as the mechanisms used to consult them. Its main role is to control access to the sources. The documents located in the workspace of the MEs, or those that are part of a product, such as the documentation of the system or the 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 others to request the documents.

3.3 Agents Collaboration

As we mentioned before, agents must collaborate with others in order to complete their jobs. In this section we present an example of how this occurs. The example shows how the staff agent and the KMA communicate between them to support the ME in the search of knowledge sources that can help him/her. As Figure 3 shows, the staff agent captures each event that is trigged by the ME on the graphical user interface (GUI).

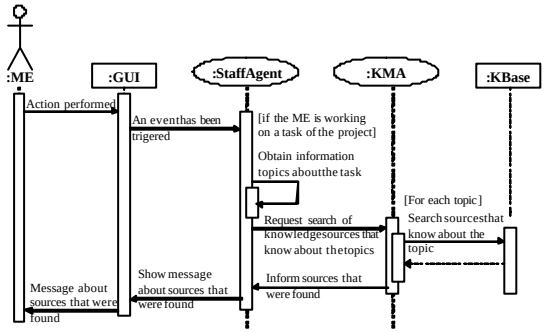

Figure 3. The staff agent and the KMA communicate each other to help the ME search knowledge sources relevant to the problem at hand

First, the ME sees a list of the projects s/he is assigned. These are shown by the staff agent through its GUI. When an ME selects one project, an event is triggered and captured by the staff agent, which obtains the information of the project, identifies knowledge topics (system and module where the problem appeared, kind of problem, etc.) and generates some rules to request the KMA to search for knowledge sources. To create the rules, the staff agent tries to identify the knowledge that the engineer would need to carry out the assignment. Also the agent considers the types of sources the engineer consults, assigning more relevance to the sources that the engineer consults most frequently. The KMA searches the knowledge base for knowledge sources that can have information related to the topics of the task to be perfomed. Once the KMA finds the sources, it informs the staff agent about them. Next, the staff agent notifies the ME of the relevant sources that were found.

Finally, if the ME wants to see the sources found, s/he chooses a button in the staff agent window, and the agent will display a window with the list of sources grouped by kind (see Figure 4). When the maintainer selects one source from the list, the window shows some information related to that source, such as its location and the knowledge it has.

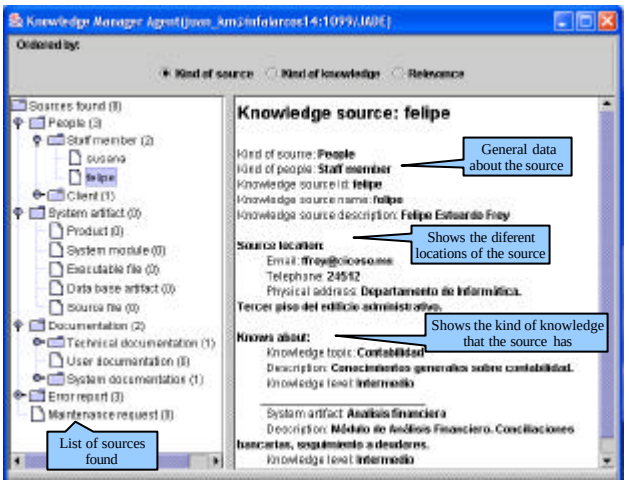

Figure 4. Window that shows the list of knowledge sources found

3.4 Some Aspects of Implementation

The platform chosen to implement the multiagent system is JADE [3] which is a FIPA compliant agent platform, implemented in Java and developed as an open source project. JADE has been used in the development of other systems in the domain of knowledge management [1, 4, 9, 11, 19].

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 repository where more generic knowledge is stored. The data are classified following an ontology for software maintenance proposed in [22], which is an extension of that of [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 companies generate huge amount of knowledge that should be stored and processed. In this way, companies would obtain more benefit from it. This paper presents the architecture of a multiagent system in charge of storing and managing information, expertise and lessons learned which are generated during the software maintenance process. The multi agent architecture facilitates the reuse of good solutions and the sharing of lessons learned. Thereby, the costs of maintenance in time and effort should decrease.

As future work we are planning to carry out a case study in a real maintenance company in order to evaluate the interaction between the staff agent and the maintainers.

ACKNOWLEDGEMENTS

This work is partially supported by the MAS project (grant number TIC2003-02737-C02-02), Ministerio de Ciencia y Tecnología, SPAIN, and by CONACYT under grant C01-40799 and the scholarship 164739 provided to the first author.

REFERENCES

1. Abecker, A.; Bernardi, A. and Elst., L. (2003). Agent Technology for Distributed Organizational Memories: The Frodo Project. Proceedings of the 5th International Conference on Enterprise Information Systems. Vol. 2, pp. 3-10.
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 31: pp. 103-128.
4. Bergenti, F.; Poggi, A. and Rimassa, G. (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. (2002). Capability Maturity for Knowledge Management. 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., Chainho, P., Kearney, P., Stark, J., Evans, R., Massonet, P. (2001). Agent Oriented Analysis Using MESSAGE/UML in Agent Oriented Software Engineering, pp. 119-135.
7. Dingsoyr, T., and Conradi, R. (2002). A Survey of Case Studies of the Use of Knowledge Management in Software Engineering. International Journal of Software Engineering and Knowledge Engineering. 12 (4): pp. 391-414.
8. Dingsoyr, T., and Royrvik, E. (2003). An Empirical Study of an Informal Knowledge Repository in a Medium-Sized Software Consulting Company. In Proceedings of the 25th International Conference on Software Engineering (ICSE'2003), pp. 84-92.
9. Gandon, F. (2002). A Multi-Agent Architecture For Distributed Corporate Memories. Proceedings of the Sixteenth European Meeting on Cybernetics and Systems Research.
10. Kitchenham, B.A., Travassos, G.H., Mayrhauser, A., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R. and Yang, H. (1999). Towards an Ontology of Software Maintenance. Journal of Software Maintenance: Research and Practice. 11, pp. 365-389.
11. Knowledge On Demand (KOD), IST Project, IST-1999-12503, http://kod.iti.gr/, http://www.kodweb.org.
12. Lindvall, M., and Rus, I. (2003). Knowledge Management for Software Organizations. In: Managing Software Engineering Knowledge. Aurum, A., R. Jeffery, C. Wohlin and M. Handzic (eds.). Berlin. Springer. pp. 73-94.
13. Macintosh, A. (1997). Position paper on Knowledge Asset Management http://www.ntgi.net/ntgi/y2k/kmfr.html
14. Meeham, B., and Richardson, I. (2003). Identification of Software Process Knowledge Management. Software Process Improvement and Practice, pp. 45-55.
15. Mercer, S., and Greenwood, S. (2001) A Multi-Agent Architecture for Knowledge Sharing. Proceedings of the Sixteenth European Meeting on Cybernetics and Systems Research.
16. Nebus, J. (2001). Framing the Knowledge Search Problem: Whom Do We Contact, and Why Do We Contact Them?. Academy of Management Best Papers Proceedings, pp. h1-h7.
17. Oliveira, K.M, Anquetil, N., Dias; M.G, Ramal, M., Meneses, R. (2003). Knowledge for Software Maintenance. Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'03), San Francisco, 1-3 July, pp. 61-68.

18. Pigoski, T.M. (1997): Practical Software Maintenance. Best Practices for Managing Your Investment. Ed. John Wiley & Sons, USA.
19. Poggi, A.; Rimassa, G. and Turci, P. (2002). An Intranet Based Muti-Agent System for Corporate Memory Management. Proceedings of the Sixteenth European Meeting on Cybernetics and Systems Research.
20. Polo, M., Piattini, M., and Ruiz, F. (2002). Using a Qualitative Research Method for Building A Software Maintenance Methodology. In Software Practice & Experience. John Wiley and Sons. 32 (13): pp. 1239-1260.
21. Rodriguez, O. M., A. I. Martínez, et al. (2004). Understanding and Supporting Knowledge Flows in Community of Software Developers. X International Workshop on Groupware, San Carlos (Costa Rica) (accepted to publish) .
22. Ruiz, F., Vizcaíno, A., Piattini, M. and García, F. (2004). An Ontology for the Management of Software Maintenance Projects. To be published in the International Journal *of Software Engineering and Knowledge Engineering*.
23. Szulanski, G. (1994). Intra-Firm Transfer of Best Practices Project. *American Productivity and Quality Centre*, Houston, Texas, pp. 2-19.
24. Tacla, C., and Barthès, J-P. (2002). A Multi-Agent Architecture for Knowledge Management System. Second IEEE International Symposium on Advanced Distributed Computing Systems. ISADS.
25. Vizcaino, A., Favela, J., Piattini, M., García, F. (2003). Supporting Software Maintenance in Web Repositories through a Multi-Agent System. In Menasalvas, E., Segovia, J., and Szczepaniak , P. S. (Eds.) First International Atlantic Web Intelligence Conference (AWIC'2003). LNAI 2663, pp. 307-317.