Understanding and Supporting Knowledge Flows in a Community of Software Developers

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

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

Abstract. Knowledge sharing is a collective process where the people involved collaborate with others in order to learn from them. This effort creates communities where each member cooperates by sharing knowledge about a common domain. An example of these kinds of communities is software maintenance groups, where their members must collaborate with others, and share their knowledge and experience in order to complete their assignments. This paper presents a study carried out in two software maintenance groups to understand how the knowledge flows through these groups, that is, how their members share their knowledge when they perform their activities. The approach used to model the flows of knowledge and to identify the problems that affect that flow are described, as well as the main problems detected, and how we are trying to solve them with an agent-based knowledge management system.

1 Introduction

Nowadays, knowledge has becoming a very important factor for organizations' competitive advantage. In fact, intellectual capital is one of the most important assets for many organizations [9]. Therefore, it is important to increment the sharing of knowledge through their communities, since this can increase their intellectual capital and their performance [10].

Knowledge sharing is a collective process where the people involved collaborate with others in order to learn from them [9]. This effort creates communities where each member cooperates by sharing knowledge about a common domain. These are called communities of practice [9, 10].

An example of communities of practice is software maintenance groups. Software maintenance is a knowledge intensive work where maintainers must collaborate with other members of the team, and share their knowledge and experience in order to complete their assignments. Even though, maintainers mainly use their own experience, generally they do not have enough knowledge to complete their work and need to consult other sources of information such as other members of the team in order to finish some assignments [21]. However, this could be a difficult task because those sources are often limited, inaccessible, or unknown [17].

Knowledge management provides methods and techniques that can help software organizations to increment the collaboration of their members; for example, by sup-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 52–66, 2004.

porting the sharing of knowledge between them. Even though this could bring many benefits to software organizations, little work has been done to apply it in the software maintenance process [16].

We think that providing software maintenance groups with tools that facilitate their members to collaborate and share their knowledge, the performance of these communities can be incremented. However, before these tools can be developed, some questions must be answered [1], such as: what kinds of problems could be solved?, what is the knowledge involved in the activities performed by the maintenance groups?, and how they share that knowledge?.

In order to answer the above questions, we conducted two case studies in two software maintenance groups. The case studies were focused on identifying how the knowledge flows through the teams, what are the problems that affect these flows and how we can address them with a knowledge management tool. This paper presents this work. The content is organized as follows: section 2 introduces some theoretical background about knowledge and *knowledge flow*. After that, section 3 presents our approach for modelling *knowledge flows* in cooperative communities, and how scenarios can be used to show the problems that affect the flow of knowledge. In section 4 some findings of the study are discussed. Then, the work related to this research is in section 5 and section 6 presents a multi-agent knowledge management system that was designed and implemented from the results of the case studies to promote collaboration between the members of a maintenance group. Finally the conclusions of this work are in section 7.

2 Theoretical Background

There are different conceptions of the types of knowledge; most authors agree that knowledge could have two forms, explicit and tacit. Explicit knowledge can be expressed in words or numbers and shared in form of data, scientific formulate, specifications, manuals, audio, video, etc. Thus, explicit knowledge is easy to share. Tacit knowledge is more personal and difficult to formalize, therefore is harder to communicate and share to others [15].

On the other hand, knowledge creation is a process where tacit knowledge becomes explicit and vice versa. These transfers between the two kinds of knowledge generate four conversion mechanisms: *socialization*, when tacit knowledge creates more tacit knowledge by people communicating with others sharing experiences; *externalization*, when tacit knowledge becomes explicit by formalizing it in documents, reports, etc.; *combination*, when explicit knowledge creates more explicit knowledge by combining information that resides in formal sources like documents, reports, etc.; and *internalization*, when explicit knowledge creates tacit, for example by consulting formal sources of information to obtain knowledge [15].

When people need to do some activity, they mostly use their own knowledge and experience, but when this is not enough, they use different techniques to consult other sources to obtain information that will help them to make the decisions required [4]. These sources could be documents, or other members of the community.

As we stated before, these communities of people that collaborate to share their knowledge and learn together are called communities of practice [9, 10]. In order to observe how the knowledge flows through the communities of practice of an organi-

zation, it is important to identify the knowledge they need and how they use it and obtain it while they perform their activities and make decisions. Next we describe how this process was done in the case studies carried out.

3 The Study

We conducted two case studies in two software maintenance groups. The first group was from a department in charge of the development and maintenance of the software systems used to the management of a scientific research institution. This department maintains applications of 5 domain areas: finances, human and material resources, academic productivity, and services for students. The second group was from a company that develops and maintains software for the management of telephone services. This company has more than 4000 clients to whom it provides services.

The studies were performed based on interviews, observation and analysis of documentation. All the interviews were recorded for later analysis. We also performed a bibliographic research to compare our findings with those that have been described in the literature.

The methodology followed in the study has four main steps: in stage 1 the main sources of information were identified and classified (documents and people); then, in stage 2, the knowledge that those sources had was defined and classified; in the third stage the processes and activities performed by the group were modelled to identify the people involved, how they collaborate to fulfil their tasks, and how the knowledge and sources interact in those activities; finally, in stage 4 the main problems that can affect the flow of knowledge were highlighted through the definition of scenarios. In the next section, the approaches used to identify the *knowledge flows* and the problems that affect these flows are described.

3.1 Identifying *Knowledge Flows* in a Software Maintenance Group

The identification of the *knowledge flows* in the software maintenance groups was based on the generic model showed in Figure 1. The model considers that, when a maintainer must perform some activity, s/he uses her/his knowledge and experience to analyze the problem and find possible solutions. Frequently, maintainers do not have all the knowledge they need, so they consult other sources of information to obtain it [21]; then, they use all these knowledge to decide which solution to implement. The processes of analyzing the problem and making decisions generate new knowledge and experience for the maintainers that can be shared later with the rest of the team. They obtain and share knowledge by using the conversion mechanisms of the knowledge creation process [15].

The conceptual model enables the identification of the knowledge involved in the activities performed by the members of the team. Here, the use of a graphic process modelling technique could be very useful [13]. An example of the latter is presented in Figure 2, which shows the main activities performed in the definition of the modification plan performed for one of the groups studied. The model also shows the people involved in those activities, the knowledge they have together with their relevance to the activities modelled, and the main sources used, created or modified in the activities.

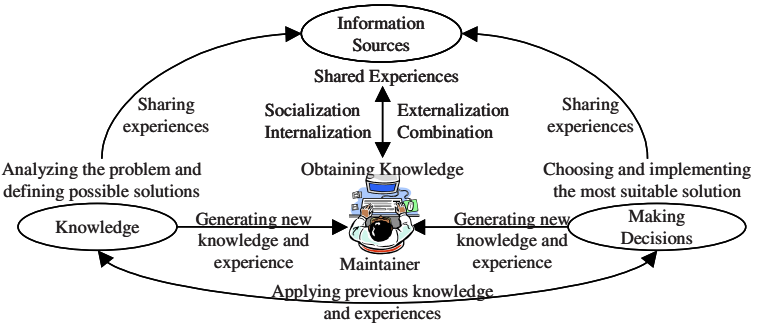

Fig. 1. This shows the generic conceptual model of the *knowledge flow* in a process decision making.

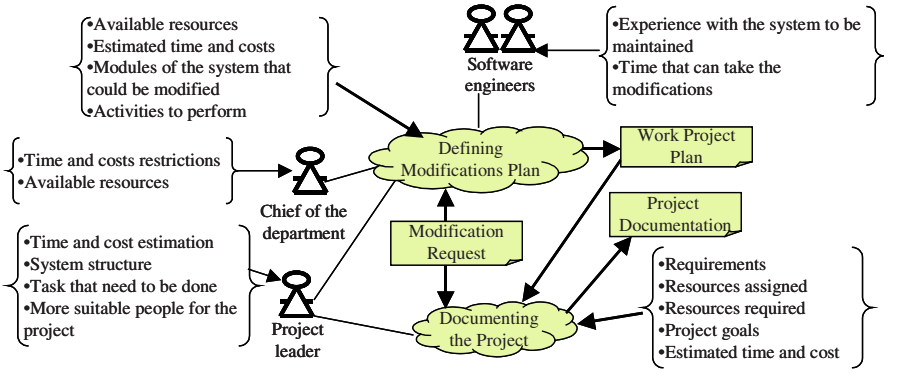

Fig. 2. This is an example of a model of activities performed by members of a maintenance group.

Once the activities have been modelled, the next step is to define the decisions that must be made by the people involved. To do that, we used the schema showed in Table 1. This schema helps to identify the knowledge that people in charge of the activities must have to make the decisions needed to fulfil their activities, and the sources of information they consult to help them making those decisions. At this step, it is important to identify the mechanisms that people can use to consult the sources, as well as those used to share the knowledge generated from the development of the activities; for example, the documentation of the plan of modifications of Figure 2.

The analysis of the activities performed by the maintainers, using the graphical model and the information from the tables, facilitates to identify when they need to consult other members to obtain information, and when they must collaborate and share their knowledge. The identification of the mechanisms that the community uses to share knowledge helps to understand how the knowledge actually flows through the community. Finally, all these can be used to identify what kinds of tools might be useful to increment the flow of knowledge in the team.

Table 1. Schema used to identify knowledge in decision making.

Role	Project leader	
Activity	To define modification plan	
Decision	To define required resources	
	To define main tasks to perform	
	To assign tasks to the participants of the project	
	To estimate the time the project would consume	
Knowledge	Previous projects experiences	
	Requirements and restrictions of the project	
	Abilities and experience of each of the possible participants of the project	
Sources of information		
Name	**Information**	**Consulted at**
Chief department	Available resources; time and cost restrictions	Telephone, Physical address, Email
Software engineers	Experience with the system that will be modified; time that could consume the modifications; time availability	Telephone, Physical address, Email
Previous projects documentation	Resources required by previous projects	Documents files, modifications logbook

3.2 Using Scenarios to Show *Knowledge Flow* Problems

The main goal of the identification of the *knowledge flow* in communities is to understand how this is occurring in order to provide mechanisms that help to increment it. Scenarios can be useful to do this, since these enable the identification of design requirements for software systems. Also, they make feasible the participation of users during the requirement specification stage [3].

A scenario is a textual description (like a history) about the activities that people might engage in while pursuing a particular concern [2]. Particularly we identified and defined scenarios that illustrate the problems that affect the flow of knowledge in the groups studied. These scenarios were classified in two main problems. The first one is related to the management of information and knowledge sources; for example, finding experts, document management, etc. The second one is related to the management of lessons learned; for example, experiences reuse to estimate the resources that a project might consume, to identify the modules and source files that might be changed in a modification request, to identify other modules of the system that could be affected by the changes, etc. Next we present two examples of scenarios, one of each kind of problem.

Expert Finding (Information Sources' Management). Mary is a software engineer that must make some changes in the finances system. Since her knowledge in the domain of finances is not good enough, the changes to the system are taking more than a week of the estimated time. At the end of the week, Susan, the chief of the department, while she was checking the advances of the project, she detects the delay and asks Mary the reasons of that delay. Mary tells Susan the problem and since Susan has experience with finances, she tells Mary how the problem could be solved. Finally, Mary solves the problem the same day.

Identifying Files Affected by the Changes (Lessons Learned). While Tom was performing changes in the structure of a table of the data base of one of the systems maintained by the team, he observed that there were some files of the system that

would also have to be changed. Because of this, Tom wrote a report in order to not forget what modules he would have to change if he had to modify that table in the future. Later, Tom left the team, and Mike took his place. One day, Mike needed to modify that table but, because he did not know about the report made by Tom, he did not consult it, and when he made the changes and tested them, he thought that everything was fine. Then, when the system was executed, two modules started to fail, as Tom had specified in the report before.

Discussion. As can be seen from the scenarios, there was knowledge in the team that was not used because of ignorance of the people who could have benefited with it. This fact has already been commented on by other authors, such as Szulanski [18] 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. Thus, it is important to solve this problem in order to increase the *knowledge flow* in the community group. In next section, some of the findings obtained from the analysis of the models and scenarios defined in the case studies carried out are discussed.

4 Findings' Discussion

Our interest has focused on understanding how the knowledge flows in the community in order to provide mechanisms to increment that flow. In the study, we found tree main aspects that we consider important to the flow of knowledge. These findings are discussed in this section; also some fragments of the interviews are presented to support our discussion.

1) Team Work. Team work is a good technique to promote the sharing of knowledge in the group. When people must collaborate to accomplish one task, they share their knowledge by discussing and making decisions together.

> *when I started working here, I had to work with a [partner], he knew about the tools, and I knew about the procedures in the domain area we were working on, I had more experience in some programs and he in others, so we share the work in that way*

When the result of the job of one member of the community is needed by others, they have to find ways to share information.

> *I used to use the logbook because my work was related to others. The secretary took the calls and wrote the problems in the logbook, then she sent it to me, I solved it, and I documented it in the logbook.*

In contrast, when people work alone, they usually do not share knowledge with others.

> *but now, ... my work is not related to almost anybody, so, I am not documenting in the logbook ... as I now remain almost isolated!*

These remark the importance to promote team working in order to increase the collaboration and sharing of knowledge through the members of a community. An ex-

ample of how to do this in software maintenance communities is by implementing pair programming techniques [14].

2) Experience Reuse. Lessons learned are a good way for knowledge and experience reuse. We found that lessons learned could be useful, for example, to identify the modules that need to be changed and those that could be affected by the changes. Moreover, past experiences help the members of the team to make estimations about the resources that the modifications can consume. Therefore, it is important to identify and capture these lessons learned in order to make them available to the team.

> *the logbook helps me to do changes that aren't done every day, but once or twice a year. If I don't know very well what to do, then, I look at the logbook, I see how was done before, and then I do it*

3) Know What the Team Knows. Knowing which is the knowledge of the group is important for knowledge sharing, if people do not know what knowledge is available in some way to consult, this will be lost and never used.

> *but, how she could know (what I knew) it if I didn't tell her before*

It is important to provide tools that enable to the members of the team to know what sources of information they can consult and what kind of knowledge or information they can obtain from those information sources. This is one of the main problems that knowledge management tools must solve [16]. One approach that can help to solve this problem is to use software agents. Agents can act like a personal assistant that knows the engineer's profile and identifies the needs of knowledge and search for sources that can help the engineer to fulfill his/her job [11]. Based on this, we have designed a multi-agent based knowledge management system for software maintenance. This is focused on supporting experience reuse and helping to know what the group knows enabling the collaboration between the members of the team. In the next section we will describe briefly the related work in this area, to continue next with the description of the system.

5 Related Work

Agents have characteristics that make them a good alternative for developing systems for supporting knowledge and information management in collaborative communities [19]. For example, Guizzardi et al. [7, 8] have proposed the use of agent oriented methodologies for analyzing and designing systems for supporting collaborative learning communities such are communities of practice; they have designed an agent-based architecture to support collaborative learning in education [6]. There are other approaches that can be useful to tackle some of the problems identified in the case studies carried out for this work; for example, Jasper [5], a multi-agent system which agents can generate profiles of their users while they consult information, and inform to other agents about sources found which can be relevant to the users of these last agents. However, these approaches are too general and do not address some specific problems of software maintenance groups; for instance, to identify which knowledge is needed by the maintainer for solving a particular maintenance request. To address this problem, it is not sufficient to know the engineer's profile; it is also required to know about the context of the activities the engineer must perform in order to search

for information sources that can help the maintainer to fulfill his/her job. To accomplish this, they should not be used only as an external application for searching into a knowledge or information repository when the maintainers decides s/he needs information about something, agents must be integrated into the work environment of the maintainers, for example, providing support for managing the projects and tasks where maintainers are working on.

On the other hand, there are few works that use agents to support knowledge management in software maintenance; Vivacqua [20] developed an agent based expertise locator which has been applied in a software maintenance domain, in this system users must define their knowledge profiles; this is one of the problems why these kinds of systems are not well accepted for software organizations [1]; we think it is important that these profiles can be generated and managed automatically by the agents. Mercer and Greenwood [12] propose a multi-agent architecture for knowledge sharing focused on helping to develop good programming practices.

As can be seen, there are different works that address different aspects of knowledge management needs in collaborative communities. However, they are not directly oriented to support *knowledge flows* for a particular community need as the found in our case with software maintenance groups. In the next section we describe the system developed.

6 An Agent-Based Knowledge Management System for Software Maintenance

There are several reasons why agents are good technical alternative for the development of knowledge management software [19]. First of all, agents are proactive; this means they act automatically when it is necessary. One of the obstacles to implementing knowledge management in software organizations is that employees do not have time to introduce or search for knowledge [16]. During their daily work, different kinds of knowledge and experiences are created, but not captured in a formal representation; they are only in the engineer's head. To increment the *knowledge flow* in software maintenance, it is important to address this problem without increasing the maintainer's work. Agents can capture and manage information automatically; for example, acting like a personal assistant [11].

Moreover, agents can manage both distributed and local knowledge. This is an important feature since the software maintenance knowledge is generated by different sources and often from different places.

Another important issue is that agents can learn from their own experience. Consequently, the system is expected to become more efficient with time since the agents have learnt from their previous mistakes and successes [11].

Finally, in a multi-agent system each agent may use different reasoning techniques depending on the situation. For instance, they can apply ID3 algorithms to learn from previous experiences and case-based reasoning to advise a client how to solve a problem.

Based on the problems detected in the case studies, and on the agents' characteristics to facilitate the solution of some of the problems, we have developed a prototype of a multi-agent knowledge management system. The preliminary tests made to the prototype had showed that it can help to promote the collaboration and sharing of knowledge between the members of a maintenance group; this is by informing to the

maintainers who has knowledge that can be relevant to the activities they are performing. Next the multi-agent architecture of the system is presented; also a scenario that shows how the system can help to promote collaboration and sharing of knowledge is described.

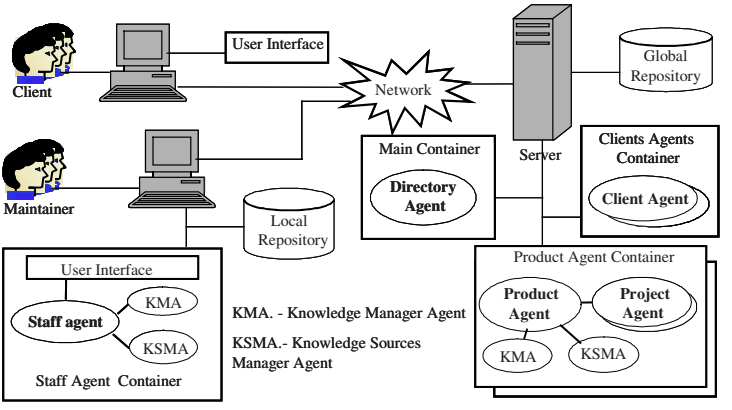

Fig. 3. Multi-agent architecture of the system.

6.1 Multi-agent Architecture

The system's architecture has five main types of agents (see Figure 3). Four of them represent the main elements identified in the case studies carried out. Next we describe each agent.

The *staff agent* is a mediator between the maintainer and the system. It acts like an assistant to him. This agent monitors the maintainer's activities and requests the KMA to search for knowledge sources that can help him to perform his/her job. This agent has information that could be used to identify the maintainer profile, such as which kinds of knowledge s/he has or which kinds of sources s/he often consults.

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.

Each maintenance project is managed by a *project agent*, which is in charge of informing the maintainers involved in a project about the tasks that they should perform. This is done through the staff agent. The project agents also control the evolution of the projects.

The *client agent* manages information related to the maintenance requests 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.

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.

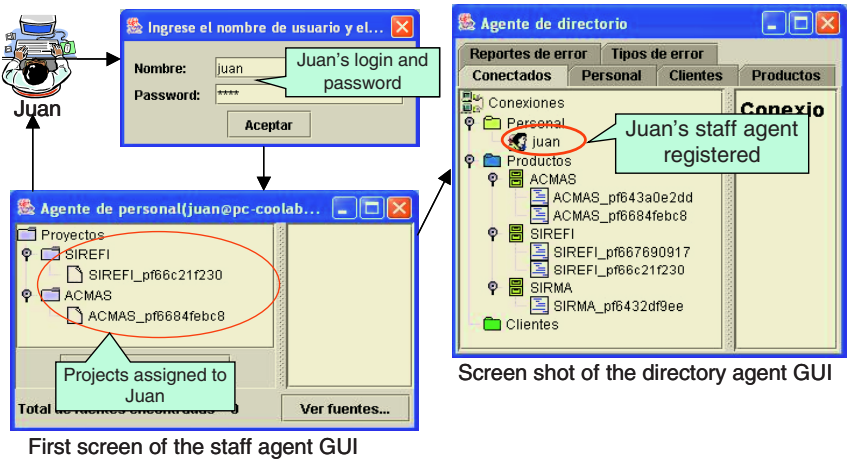

Fig. 4. Screen shots showing the user registration in the system.

The architecture also has two types of auxiliary agents: The Knowledge Manager Agent (KMA) and the Knowledge Sources Manager Agent (KSMA). The KMAs are in charge of the management of the knowledge base. These kinds of agents provide support by capturing and searching for knowledge in the knowledge base. The KSMAs have control over the knowledge sources, such as electronic documents. These agents know the physical location of those sources, as well as the mechanisms used to consult them. Their main role is to control access to the sources.

6.2 Promoting Collaboration and Sharing of Knowledge: A Scenario of Use

To exemplify how the tool helps to promote the collaboration and sharing of knowledge in the maintenance groups, we are going to describe a scenario of use.

Juan is a maintainer assigned to two modification projects. When Juan starts the system and provides his login and password, his staff agent is registered by the directory agent, and then it shows a window with the list of projects assigned to Juan (see Figure 4). This information is provided by the projects' agents in charge of those projects.

Juan decides to solve one of those projects which is a problem reported by a client. When Juan presses the corresponding button, the staff agent shows a window with the information of the problem report and generates some rules to request the KMA to search for knowledge sources that can help Juan to solve that problem. 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. When the search has finished, the KMA sends a message to the staff agent informing it about the sources found. The staff agent displays a message with the number of sources found in order to inform the engineer (see figure 5).

Juan decides to look for the sources found, so he chooses the corresponding button in the staff agent GUI, and the agent displays a window with the list of sources grouped by categories. Then, Juan selects an item of the sources list which corre-

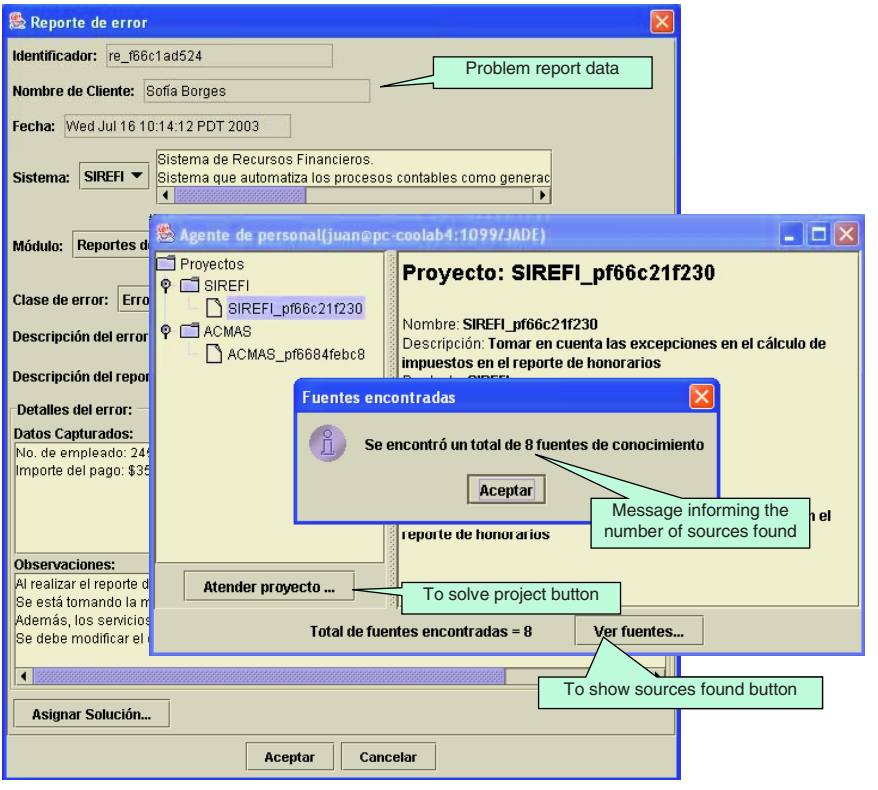

Fig. 5. Screen shot that shows how the system informs to the user the total of knowledge sources found.

sponds to a staff member called Felipe, and the agent shows some information related to Felipe: knowledge that he has, location, etc. (see Figure 6). Juan looks for the knowledge that Felipe has, and finds out that Felipe could help him to solve the problem, since he has experience in the domain area of the system where Juan must do the changes. Because of this, Juan decides to consult Felipe calling him to the telephone number provided by the tool. Finally, Felipe brings Juan some advices that help him to solve the problem.

As can be seen in the scenario just described, the tool provided support to Juan to find a colleague that had the knowledge to help him solve the problem. As a consequence Juan consulted Felipe, who gave Juan some advices that facilitated him to solve the problem. Thereby, the tool helped to promote the collaboration and sharing of knowledge between Juan and Felipe. A similar scenario was identified in the case studies carried out, but, in that case, the maintainer did not consult his colleague at time because he did not know that the colleague had knowledge that could be useful to help him to do the changes.

The tool not only shows information about other colleges, it also presents different kinds of sources of information, such as documents, previous problem reports or maintenance requests which could be related with the project the maintainer must

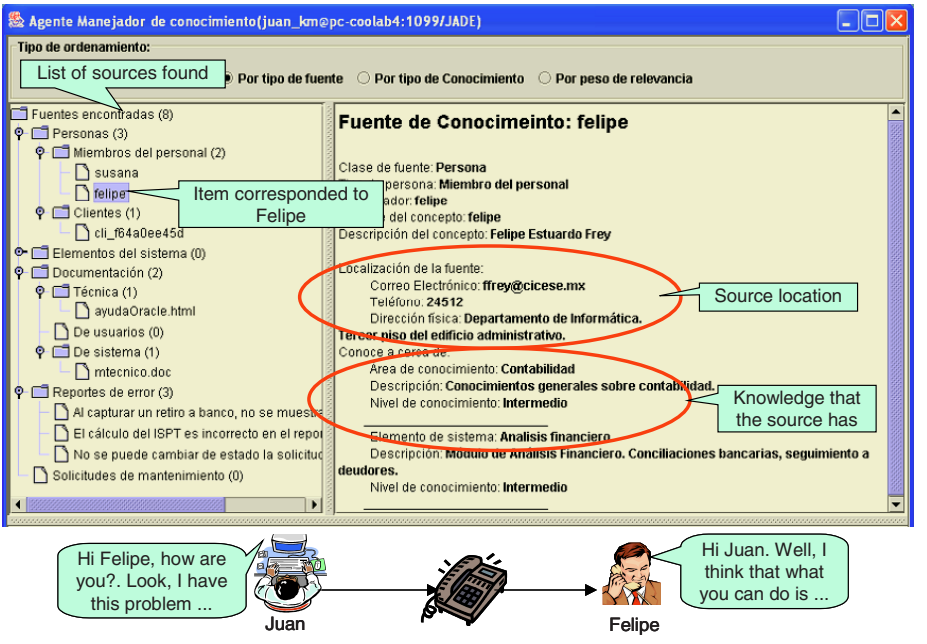

Fig. 6. Screen shot showing the list of sources found; we can observe that Felipe is one of these knowledge sources.

solve. Therefore, the tool helps maintainers to consult sources where s/he can obtain information or knowledge to help him/her to make decisions required by the activities s/he must carry out. Moreover, the tool provides an option for capturing the solution or the changes made to the system by the maintainer, thus, these information can be later used for others to solve similar problems. In other words, the tool enables maintainers to consult information sources to obtain knowledge that can be helpful to make decisions about the tasks they must carry out, and also to share the knowledge generated by making those decisions by capturing the solutions and changes made by maintainers (see Figure 7), in order to make them available for others; a process similar to the *knowledge flow* model illustrated in Figure 1.

7 Conclusions and Future Work

In this paper we presented a study carried out to understand how the knowledge flows through two communities of practice in software maintenance. The study focused on identifying the knowledge needed by the members of the group while they perform some tasks and make decisions, and how they obtain and share that knowledge. To do this, we used some qualitative techniques and proposed an approach for modeling the flow of knowledge in software maintenance teams.

On the other hand, the study has provided some aspects that must be considered to increase the flow of knowledge in the groups studied; for example, help them to know the knowledge they can consult and where they can find it. We think that software

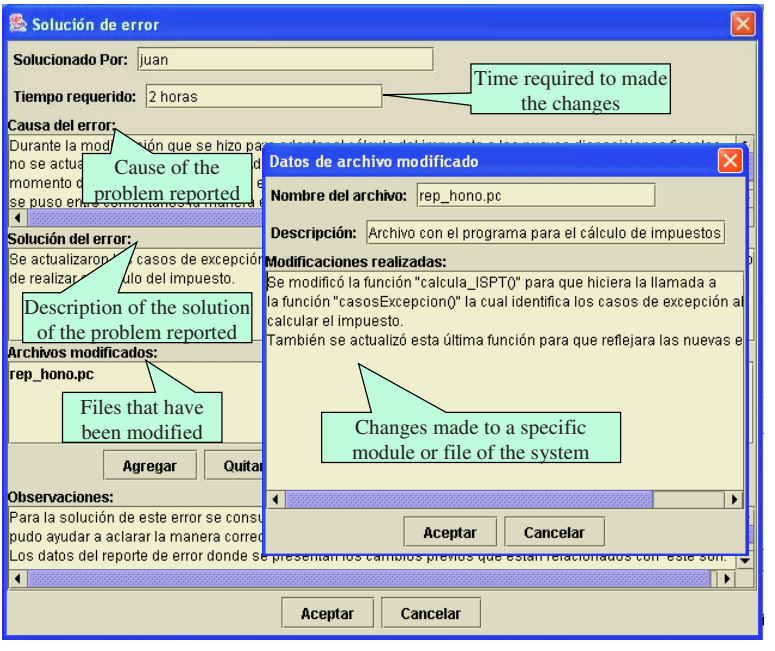

Fig. 7. Example of a solution of a problem reported and the description of the changes made in a modified file.

agents can help to solve some of these problems. Thus, we have designed an agent based knowledge management system for software maintenance that helps to solve some of the problems detected in the studies, and at the same time, helps to increment the flow of knowledge in the software maintenance groups, by promoting the collaboration and sharing of knowledge between their members. The basic characteristics of this system have been obtained from the results of the study. A first prototype has been developed and tested with scenarios that show how a tool with these characteristics can address some of the problems detected in the case studies carried out.

At the moment we are working on improving the methodology to provide more and better information for designing agent-based knowledge management systems for supporting *knowledge flows* in communities of practice in software organizations. With the information obtained by applying the improved methodology, we are planning to improve the system's prototype, in order to perform another case study to evaluate how the tool is perceived by software maintainers. To do this, some aspects must be considered first, such as how the tool can be integrated into the maintainer's environment.

Acknowledgements

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

References

1. Aurum, A., Jeffery, R., Wohlin, C., Handzic, M., Preface, in Managing Software Engineering Knowledge, Aurum, A., et al., (eds.), Springer, Berlin, (2003), ix-xv.
2. Carroll, J. M., Rosson, M. B., Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario, ACM Transactions on Information Systems, 10(2), (1992), 181-212.
3. Chin, G. J., Rosson, M. B., Carroll, J. M., Participatory Analysis: Shared Development of Requirements from Scenarios, in Conference on Human Factors in Computing Systems (CHI97), Atlanta, GA, USA, ACM/SIGCHI, (1997), 162-169.
4. Choo, C. W., La Organización Inteligente: el Empleo de la Información para dar Significado, Crear Conocimiento y Tomar Decisiones, Oxford University Press, Oxford, USA, (1999).
5. Davies, J., Weeks, R., Revett, M., Jasper: Communicating Information Agents for WWW, presented at the 4th International Conference on the World Wide Web, Boston, USA, (1995), available at http://www.w3j.com/1/davies.180/paper/180.html, last visited 23-06-2004.
6. Guizzardi, R. S. S., Aroyo, L., Wagner, G., Agent-oriented Knowledge Management in Learning Environments: A Peer-to-Peer Helpdesk Case Study, in Agent-Mediated Knowledge Management, Heidelberg, Springer, (2003), 15-22.
7. Guizzardi, R. S. S., Perini, A., Dignum, V., Providing Knowledge Management Support to Communities of Practice through Agent-oriented Analysis, in Proceedings of the 4th International Conference on Knowledge Management (I-KNOW'04), Granz, Austria, (2004), available at http://wwwhome.cs.utwente.nl/~souza/publications/guizzardi-perini-dignum-iknow04.pdf, last visited 23-06-2004.
8. Guizzardi, R. S. S., Perini, A., Dignum, V., Using Intentional Analysis to Model Knowledge Management Requirements in Communities of Practice, CTIT Technical Report, 03-53 University of Twente, (2003), available at http://www.ctit.utwente.nl/library/techreports/ tr03.doc/ index.html, last visited 23-06-2004.
9. Huysman, M., Wit, D. d., Knowledge Sharing in Practice, Kluwer Academic Publishers, Dordrecht, (2000).
10. Lesser, E. L., Storck, J., Communities of practice and organizational performance, IBM Systems Journal, 40(4), (2001), 831-841.
11. Maes, P., Agents that reduce work and information overload, Communications of the ACM, 37(7), (1994), 31-40.
12. Mercer, S., Greenwood, S., A Multi-Agent Architecture for Knowledge Sharing, in Third International Symposium From Agent Theory to Agent Implementation at EMCSR 2002, Vienna, Austria, (2002), available at http://www.ai.univie.ac.at/~paolo/conf/at2ai3/, last visited 23-06-2004.
13. Monk, A., Howard, S., The Rich Picture: A Tool for Reasoning about Work Context, Interactions, 5(2), (1998), 21-30.
14. Natsu, H., Favela, J., Morán, A. L., Decouchant, D., Martínez-Enriquez, A. M., Distributed Pair Programming on the Web, in Fourth Mexican International Conference on Computer Science, Tlaxcala, México, IEEE Computer Society, (2003), 81-88.
15. Nonaka, I., Takeuchi, H., The Knowledge-Creating Company, Oxford University Press, (1995).
16. Rus, I., Lindvall, M., Sinha, S. S., Knowledge Management in Software Engineering: A State of the Art Report. Data & Analysis Center for Software: ITT Industries: Rome, NY. (2001), available at http://www.dacs.dtic.mil/techs/kmse/kmse.html, last visited 23-06-2004.
17. Seaman, C., The Information Gathering Strategies of Software Maintainers, in Proceedings of the International Conference on Software Maintenance, (2002), 141-149.

18. Szulanski, G., Intra-Firm Transfer of Best Practices Project, in American Productivity and Quality Centre, Houston, Texas, (1994), 2-19.
19. van Elst, L., Dignum, V., Abecker, A., Agent-Mediated Knowledge Management, in International Symposium AMKM 2003, Stanford, CA, USA, Springer, (2003), 1-30.
20. Vivacqua, A. S., Agents for Expertise Location, in Proceedings of the AAAI Spring Symposium on Intelligent Agent in Cyberspace, Stanford, USA, AAAI Press, (1999), 9-13.
21. Walz, D. B., Elam, J. J., Curtis, B., Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration, Communications of the ACM, 36(10), (1993), 63-77.