Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach

Oscar M. Rodríguez[1], Ana I. Martínez[1], Aurora Vizcaíno[2], Jesús Favela[1], Mario Piattini[2]
[1]CICESE, Computer Science Department, México.
{orodrigu | martinea | favela}@cicese.mx
[2]University of Castilla-La Mancha, Escuela Superior de Informática, España.
{Aurora.Vizcaíno | Mario.Piattini}@uclm.es

Abstract

Software maintenance is an activity that requires lots of knowledge. For example, maintainers must know what changes should do to the software, where to do those changes and how those changes can affect other modules of the system. Frequently they do not have enough knowledge to make the best decision and must consult other information sources, but these sources are often unknown or difficult to locate. Therefore, knowledge management could be useful to address some of these problems; however, before knowledge management tools could be developed for software maintenance groups, some questions must be answered: such as what kinds of problems could be solved. In this paper a qualitative approach to the identification of knowledge management needs in software maintenance teams is presented. This approach has been applied in two case studies, and an agent-based knowledge management tool has been designed and implemented from the results obtained.

1. Introduction

Software engineering is a knowledge intensive activity [19]. Software developers must make many decisions selecting one of several choices [22]; for example, a maintainer must know what changes should do to the software, where to do those changes and how those changes can affect other modules of the system. Maintainers mostly use their own experience to do their task. However, in many occasions they do not have all the knowledge or expertise needed to complete their work and must consult other sources of information, such as the system documentation, or other collages [27]. This could be a difficult task because those sources are often limited, inaccessible, or unknown [24].

Knowledge management provides methods and techniques that help software organizations to make a better use of their knowledge [22]. For example, helping to identify who knows what, where the sources of information are and what kind of knowledge could be obtained from those sources. Even though knowledge management could bring many benefits to software organizations, little work has been done to apply it in the software maintenance process [17, 22].

The need for knowledge management in software maintenance has been observed by some authors. For example, having access to people that were involved in the development or maintenance of a system, capturing the knowledge and experience of these people, or increasing the flow of knowledge across the maintenance teams [2, 24, 25, 27]. However, in order to develop tools to support knowledge management in the software maintenance process, some questions must be answered first [1, 17], such as: what kinds of problems could be solved?, what is the knowledge involved in the activities performed by the maintenance teams?, what are the sources they consult to obtain that knowledge?, how the knowledge and sources interact in the maintenance activities?. In summary, how the knowledge flows through the software maintenance process?.

Answering the above questions is not a trivial task because the knowledge, experience, expertise, and the sources of information may vary considerably between organisations [24]. Thus, it is not sufficient to provide tools to support knowledge management in software maintenance groups, it is also important to provide mechanisms to help identifying the specific knowledge needed by them, the sources they consult to obtain that knowledge, and how these knowledge and sources are involved in the activities performed by the team.

In order to address the last goal, we have applied qualitative techniques in two case studies performed in two software maintenance groups. Particularly we have defined a methodology based on a process modelling

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

approach [7]. This paper presents the methodology we have defined, and how it has been applied to identify knowledge management needs in software maintenance groups. The content of this paper is organized as follows: In section 2, the related work to our research is discussed. In section 3 the case studies carried out and the methodology are described. Then section 4 shows how the results of the study were used to obtain design requirements for an agent-based knowledge management system. Finally, conclusions of this work are presented in section 5.

2. Related work

The related work to our research can be grouped in two categories: the identification of information sources and of knowledge in software maintenance. Next we will introduce each one.

2.1 Sources of information in software maintenance

The main work on this topic that we have found is the one proposed for Seaman [24]. Seaman conducts a research to identify the information gathering strategies used by software maintainers. She identifies the main kinds of sources of information used by maintainers when they perform their jobs. This work is a good start to the classification of the kinds of sources that could be involved in a software maintenance process.

Other works have described the kinds of documents involved in software maintenance. For example, Briand et al. [3] propose a taxonomy to classify the documents of the software maintenance process. Also there are standards and methodologies for software maintenance that define the documents required or generated during the maintenance process proposed by them [10, 18].

The previous works only mention the sources of information or documents that could be involved in the software maintenance process; they do not show how those sources are used by the maintainers while they perform their jobs, neither how to identify the knowledge that could be obtained from those sources.

2.2 Knowledge in software maintenance

Probably, program comprehension is the first area of research where the knowledge required by maintainers was studied; for example, Mayrhauser and Vans [13, 14] have studied how programmers create mental models to understand the software they must change, and what kind of knowledge is involved in those models.

More recently, work on this topic have been done from an ontological perspective, trying to identify the main concepts involved in software maintenance by defining ontologies to represent those concepts and how are they related to each other [8, 9, 11, 17, 21]. The need for an ontology in software maintenance was first introduced by Kitchenham et al. [11], they define an ontology to identify the factors affecting the results of empirical studies in software maintenance. Deridder [8] proposes that an ontology can help to link the artefacts of the development process and help maintainers to find those artefacts, and reuse them. Dias et al. [9, 17] developed an ontology to organize the knowledge used in software maintenance. Their ontology is an extension of that of Kitchenham et al., which introduces more detailed concepts involved in the maintenance process. Finally, Ruiz et al. [21] also extend the Kitchenham et al.'s ontology to define other for the management of software maintenance projects. The Ruiz et al.'s ontology considers both dynamic (i.e. workflow) and static aspects of software maintenance, while the other ontologies only consider static aspects.

The more relevant work for our research is that of Oliveira et al. [17] which was used to develop the Dias et al.'s ontology [9]. However, while these authors have oriented they research on identifying the knowledge used in software maintenance, we are also interested on finding how this knowledge flows.

Next we describe the case studies performed to observe how the knowledge flows through a software maintenance group and the methodology followed in the study.

3. The study

We conducted two case studies in two software maintenance groups with duration of five months. The first group was the 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 five domain areas: finances, human and material resources, academic productivity, and services for students. The second group was a company that develops and maintains software for telephone services management. This company have more than 4000 clients across the Mexican country.

The studies were 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. We defined two taxonomies based on our research and the previous work: one for classifying knowledge sources and the other for classifying kinds of knowledge (for

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

space limitations they are not detailed in this paper). The methodology followed in the studies is described next.

3.1 The methodology

The methodology is composed of four main steps, as it is shown in Figure 1. The process proposed to carry out these steps is an iterative one, because each stage could generate information to complement the others.

The first step starts by identifying the main documents and people involved in the maintenance process. Then the taxonomy is defined to classify the information sources found; also an ontology can be developed to help defining the relations between the sources and the other elements of the maintenance process.

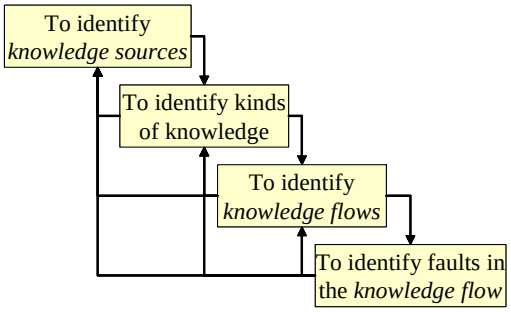

Figure 1. Steps of the methodology followed in the case studies.

The stage two starts by analysing the documents identifyed in the first step; then, the kinds of knowledge that can be obtained from the information sources found is defined together with the knowledge that the people involved in the maintenance process have or require. In this step, the taxonomy and the ontology could also help to classify the kinds of knowledge and define their relations to the other elements of the process.

In the third step a process modelling [7] approach is very useful to identify how the knowledge and sources of information are involved in the activities performed by the maintenance group. To do this, the main activities of the processes carried out by the maintainers must be identified, also the decisions they must make while they perform those activities. Then, a graphic modelling technique can be used to model these activities [15], showing people and roles involved, the knowledge required by them to perform the activities, and the sources they consult, or those that could have information to help them to fulfil their activities. These models are used to analyse how the knowledge flows through the group while they perform their activities; for example, what sources they consult to do those or what documents are generated from doing those.

Finally, in the fourth step the models are analysed to find the problems that could be affecting the flow of knowledge. For example, if the information generated from the activities is not captured, or if there are sources that could help to perform some activities, but are not consulted by the people in charge. In this stage, scenarios can help to show how these problems detected affect the *knowledge flow*, and how this could be solved. These scenarios could be used later to obtain design requirements to the development of tools to address those problems [5].

Since our main interest is to observe how the knowledge flows through a software maintenance group, also how this flow can be incremented, in this paper we focus on the stages three and four which address these goals.

3.2 Modelling *knowledge flows*

The modelling of the flow of knowledge in the maintenance groups started by identifying the roles played by the sources of information, and the knowledge and experiences of the maintainers in the decisions making process [23].

The knowledge cycle of Choo [6] proposes three fields of use of the information: 1) to create a representation of the environment, 2) to create knowledge, and 3) to make decisions; these fields are involved in a process where the shared information and experience are used to generate the knowledge needed to make decisions; then the use of this knowledge and the experience obtained by applying it in the decision making process creates new knowledge and experience that could be later shared to others.

On the other hand, as Nonaka and Takeuchi describe [16], knowledge creation is a process of interactions between explicit and tacit knowledge. Explicit knowledge can be expressed in words and numbers, and shared in form of data or documents, while tacit knowledge is more personal and hard to formalize (for example abilities and experience). The interactions between these kinds of knowledge create four conversion patterns: *socialization*, when people share tacit knowledge by interacting with others; *externalization*, when tacit knowledge becomes explicit by expressing it in formal ways; *combination*, when explicit knowledge creates more explicit knowledge by combining information that resides in formal information sources like documents, etc.; and *internalization*, when explicit knowledge creates tacit, by consulting formal sources of information to obtain knowledge [16].

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

Based on the approaches of [6] and [16], we have defined the generic model showed in Figure 2. The model considers that when a maintainer must perform an activity, s/he uses her/his knowledge and experience to analyse the problem and find possible solutions. Frequently, maintainers do not have all the knowledge they need, therefore they consult other sources of information to obtain it [27]; then, maintainers use all these to decide which solution to implement. The processes of analysing the problem and making decisions generate new knowledge and experience for the maintainers that can be later shared to the rest of the group. They obtain and share knowledge by using the conversion patterns described in [16].

Figure 2. Generic model for ***knowledge flow*** for software maintainers.

The conceptual model enables the identification of the knowledge involved in the activities performed by the members of the group. Here, a graphic process modelling technique can be very useful [15]. An example of the latter is presented in Figure 3, 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.

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 the people in charge of the activities must have to make the decisions needed, and the sources of information they consult to obtain information that helps them to make those decisions. At this step, it is important to identify the mechanisms that people can use to consult the sources; also, those used to share the knowledge generated in the activities; for example, the documentation of the modifications' plan in Figure 3.

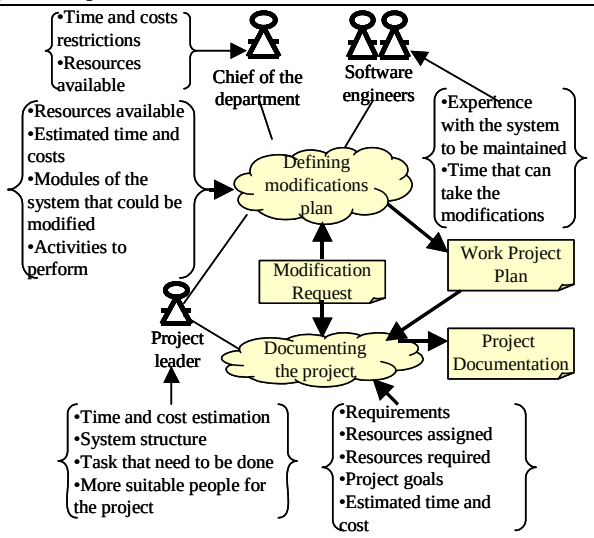

Figure 3. This illustrates a model of a modification plan definition process.

The analysis of the activities performed by the maintainers, using the graphical model and the information from the tables, are later used to understand how the knowledge flows through the team, and what techniques they use to share and obtain that knowledge. Finally this analysis can help to identify the problems that are affecting that flow, in order to provide tools to address those problems. Next we describe how the problems were identified.

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

3.3 Using scenarios to identify problems in the *knowledge flow*

Scenarios can be a useful approach to identify the problems that are affecting the flow of knowledge, and how those can be addressed. Scenarios also enable the

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

identification of design requirements for software systems and make feasible the participation of users during the requirements specification stage [5]. Particularly, the scenarios identified showed problems related to two main domains: experts finding and document management. The main problem that these scenarios highlighted was that, in many occasions people do not consult sources that could be useful to them, because they do not know about their existence, their location or the knowledge they could have; to exemplify this, next we reproduce one of the scenarios identified.

Expert finding. 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 have taken 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.

As we can observe, there was knowledge in the group that could have helped to solve the problem, but it was not used sooner because of ignorance of the people who could have benefited from it. This fact has already been commented on by other authors, such as Szulanski [26] 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 address this problem in order to increase the knowledge flow.

Software Agents are a technology that can help addresing these kinds of problems. Agents can act like personal assistants that know the engineers' profile, identify the needs of knowledge and search for knowledge sources that can help the engineer to fulfil his/her job [12]. Based on the latter and some requirements obtained from the analysis of the scenarios identified, we have developed an agent based prototype to manage knowledge; this is presented in the next section.

4. A multi-agent system to manage knowledge in software maintenance

As stated before, one of the main problems that affect the flow of knowledge is that people often do not know the sources of information that can be useful when performing some activity, therefore they do not consult them; this is one of the main challenges on applying knowledge management in software engineering [22].

Trying to solve this problem, we have developed an agent based prototype for knowledge management in software maintenance [20]. Agents have characteristics that can be useful to solve these kinds of problems; first of all, agents are proactive; this means they act automatically when it is necessary, so they can capture and manage information without direct intervention of users. For example, acting like a personal assistant [12].

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 [12].

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

Next the multi-agent architecture of the system is presented.

4.1 Multi-agent architecture

To design the multi-agent architecture, we have followed MESSAGE, a Methodology for Engineering Systems of Software Agents [4]. MESSAGE proposes different levels of analysis; however it mainly focuses on the first 2. At level 0, the system is viewed as a set of organizations that interact with resources, actors, or other organizations; its resulting model gives an overall view of the system, its environment, and its global functionality. Subsequent refinements define models at level 1, level 2 and so on. Moving from level 0 to level 1 the analysis focuses on identifying the types of agents and roles, such as it is illustrated in the next paragraphs through the application of the methodology to the definition of our architecture.

The architecture has five main types of agents: staff, product, client, project and directory agents (Figure 4).

The *staff agent* is a mediator between the maintainer and the system. It acts like an assistant to the maintainer. The rest of the agents of the system communicate with the latter through this agent. The *staff agent* monitors the maintainer's activities and requests to the *Knowledge Manager Agent* (KMA) to search for knowledge sources that can help the

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

maintainer to perform his/her job. This agent has information that could be used to identify the maintainer's profile, such as which kinds of knowledge or expertise 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.

When the *product agent* receives a maintenance request, it creates a new project and proposes the tasks that must be done in order to fulfil the request. The agent also proposes the most suitable people to perform those tasks and sends the proposal to the staff agent in charge of assisting the maintainance engineer that plays the role of project manager. The *staff agent* informs the maintainer of these proposals, and s/he decides if the proposal is accepted or modified. Once the proposal has been accepted, the project agent starts working.

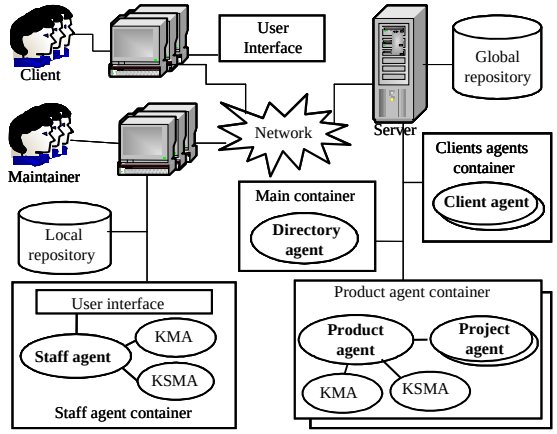

Figure 4. Multi-Agent Architecture for knowledge management in the software maintenance process.

Each 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. 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 error 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 maintainance egineers in their daily work. For example, if a maintainer is modifying a program developed in the Java language, the KMA can infer that he 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 maintainers, 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 other to request the documents.

4.2 Architecture's implementation

To evaluate the feasibility of the implementation of the architecture, we have developed a prototype. The requirements were obtained from the scenarios identified in the two case studies.

The information managed by the prototype was obtained from one of the organizations where the case studies were done. The prototype was tested specifically following the scenario described next.

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

First, the maintainer looks at a list of the projects s/he has assigned. These are shown by the *staff agent* through its screen. When the maintainer 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 do the assignment. Also the agent considers the types of sources the engineer consults, assigning more relevance to the sources that he 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 knowledge sources found in order to inform the maintainer of their availability. Finally, if the engineer wants to look for the sources, s/he chooses a button in the *staff agent* screen, and the agent will display a window with the list of sources grouped by categories (see Figure 5). When the maintainer selects one source from the list, the window shows information related to that source such as: location, the knowledge that it has, etc.

The system helps to find and locate sources of information that can be relevant to the activities performed for maintainers. In this way, sources that could not be consulted for ignorance of their existence or location could now be consulted thanks to automatic search the system does, informing to the maintainer about those sources, so that they can look if they could help them to complete their jobs.

5. Conclusions and future work

Knowledge is a crucial factor for software maintenance. Maintainers must make many decisions and need different kinds of knowledge to do it. Frequently they do not have enough knowledge to make the best decision and need to consult other sources of information. Therefore, it is important to understand how they obtain and share that knowledge, and, in general, how that knowledge flows through the software maintenance groups, in order to provide tools that help them to increase that flow.

In this paper we presented a qualitative approach to the identification of the flow of knowledge in software maintenance teams. This approach has been applied in two case studies; from these studies some problems that affect the flow of knowledge were identified. The results of the studies gave us useful insights to determine how a multi-agent knowledge management system can address some of the problems identified.

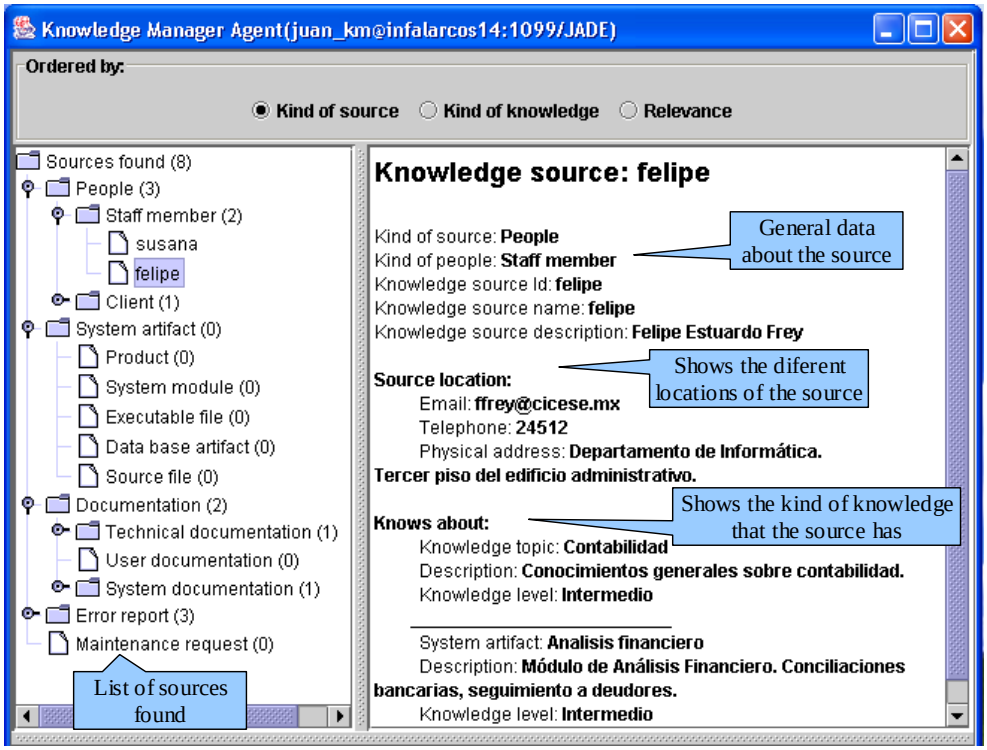

Figure 5. Screen shoot of the knowledge sources list.

Rodríguez et al., 2004, Identifying Knowledge Management Needs in Software Maintenance Groups: A qualitative approach, In: Proc. of the 5th Mexican Intl. Conf. on Computer Science (ENC 2004), Colima, México, 20-24 September, Baeza-Yates et al. (Eds.), IEEE Computer Society Press, p. 72-79

From our work, we conclude that our approach can be useful to identify knowledge management needs in software maintenance teams, and design requirements for tools to address those needs.

As future work, we are planning to conclude the knowledge management tool, and to evaluate it in a software maintenance group.

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] A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, "Preface", in Managing Software Engineering Knowledge, A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, Eds. Berlin: Springer, 2003, pp. ix-xv.

[2] K. Bennet and V. Rajlich, "Software Maintenance and Evolution: A Roadmap", In the Future of Software Engineering, Intl. Conference on Software Engineering (ICSE'2000), Limerick Ireland, 2000.

[3] L. C. Briand, V. R. Basili, Y.-M. Kim, and D. R. Squier, "A Change Analysis Process to Characterize Software Maintenance Projects", In Intl. Conference on Software Maintenance (ICSM'1994), Victoria, BC, Canada, 1994.

[4] G. Caire, W. Coulier, F. Garijo, J. Gómez, J. Pavón, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, and P. Massonet, "Agent Oriented Analysis using MESSAGE/ UML", In Agent Oriented Software Engineering, 2001.

[5] J. M. Carroll and M. B. Rosson, "Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario", ACM Transactions on Information Systems, 10(2), pp. 181-212, 1992.

[6] C. W. Choo, La Organización Inteligente: el Empleo de la Información para dar Significado, Crear Conocimiento y Tomar Decisiones, Oxford, USA: Oxford University Press, 1999.

[7] B. Curtis, M. I. Kellner, and J. Over, "Process Modeling", Communications of the ACM, 35(4), pp. 75-90, 1992.

[8] D. Deridder, "A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities", In Joint Conference on Knowledge-Based Software Engineering, Maribor, Eslovenia, 2002.

[9] M. G. B. Dias, N. Anquetil, and K. M. d. Oliveira, "Organizing the Knowledge Used in Software Maintenance", Journal of Universal Computer Science, 9(7), pp. 641-658, 2003.

[10] ISO/IEC, "ISO/IEC FDIS 14764:1999, Software Engineering-Software Maintenance", Secretariat: Standard Council of Canada, Standard 1999.

[11] B. A. Kitchenham, G. H. Travassos, A. v. Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, and H. Yang, "Towards an ontology of software maintenance", Journal of Software Maintenance: Research and Practice, 11(6), pp. 365-389, 1999.

[12] P. Maes, "Agents that reduce work and information overload", Communications of the ACM, 37(7), pp. 31-40, 1994.

[13] A. v. Mayrhauser and A. M. Vans, "Program Comprehension During Software Maintenance and Evolution", IEEE Computer, 28(8), pp. 44-55, 1995.

[14] A. v. Mayrhauser and A. M. Vans, "Identification of Dynamic Comprehension Processes During Large Scale Maintenance", IEEE Transactions on Software Engineering, 22(6), pp. 424-437, 1996.

[15] A. Monk and S. Howard, "The Rich Picture: A Tool for Reasoning About Work Context", Interactions, 5(2), pp. 21-30, 1998.

[16] I. Nonaka and H. Takeuchi, The Knowledge-Creating Company, Oxford University Press, 1995.

[17] K. M. d. Oliveira, N. Anquetil, M. G. B. Dias, M. Ramal, and R. Meneses, "Knowledge for Software Maintenance", In Proc. of the 15th Intl. Conference on Software Engineering and Knowledge Engineering, San Francisco, California, USA, 2003.

[18] M. Polo, M. Piattini, and F. Ruiz, "Using a Qualitative Research Method for Building a Software Maintenance Methodology", Software Practice & Experience, 32(13), pp. 1239-1260, 2002.

[19] O. N. Robillard, "The Role of Knowledge in Software Development", Communications of the ACM, 42(1), pp. 87-92, 1999.

[20] O. M. Rodriguez, A. Vizcaino, A. I. Martínez, M. Piattini, and J. Favela, "Using a Multi-Agent Architecture to Manage Knowledge in the Software Maintenance Process", In Intl. Conference on Knowledge-Based Intelligent Information & Engineering Systems, Wellington, New Zealand, 2004.

[21] F. Ruiz, A. Vizcaíno, M. Piattini, and F. García, "An Ontology for the Management of Software Maintenance Projects", Intl. Journal of Software Engineering and Knowledge Engineering, accepted for publication, 2004.

[22] I. Rus, M. Lindvall, and S. S. Sinha, "Knowledge Management in Software Engineering: A State of the Art Report", Data & Analysis Center for Software: ITT Industries, Rome, NY, 2001.

[23] V. L. Sauter, "Intuitive Decision-Making", Communications of the ACM, 42(6), pp. 109-115, 1999.

[24] C. Seaman, "The Information Gathering Strategies of Software Maintainers", In Proc. of the Intl. Conference on Software Maintenance, 2002.

[25] J. Singer, "Practices of Software Maintenance", In Proc. of the Intl. Conference on Software Maintenance, 1998.

[26] G. Szulanski, "Intra-Firm Transfer of Best Practices Project", In American Productivity and Quality Centre, Houston, Texas, 1994.

[27] D. B. Walz, J. J. Elam, and B. Curtis, "Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration", Communications of the ACM, 36(10), pp. 63-77, 1993.