A Multi-Agent System for Expertise Localization in Software Development

José Ramón Martínez García[1], Ramón René Palacio Cinco[2], Joel Antonio Trejo-Sánchez[3]
Luis-Felipe Rodríguez[1], Joaquin Cortez[1]

[1]Instituto Tecnológico de Sonora, Unidad Nainari,
Antonio Caso 2266, Ciudad Obregón, Sonora, México.
e-mail:joseramonmg26@gmail.com, luis.rodriguez@itson.edu.mx
joaquin.cortez@itson.edu.mx

[2]Instituto Tecnológico de Sonora, Unidad Navojoa,
Ramón Corona S/N, Colonia ITSON, Navojoa Sonora, México.
e-mail:ramon.palacio@itson.edu.mx

[3]Universidad del Caribe, Dpto. Ciencias Básicas e Ingeniería,
Esquina Fraccionamiento Tabachines, Cancún, Q.R.
e-mail:jtrejo@ucaribe.edu.mx

***Abstract*— In software industry, *expertise* is fundamental to the timely ending of projects, since tasks are solved in a more efficient manner. The *expertise* of an organization exists in people or artifacts and it is not easy to identify when required because this information does not reside in a repository to facilitate its management in terms of storage, consultation and distribution. This causes uncertainty among members of the organization to determine the appropriate *expertise* to solve a project activity. The aim of this paper is to present a multi-agent system that supports the *expertise* location in software development. Using a knowledge flow methodology the barriers that prevent the flow of knowledge and interaction in software development activities were identified and using this information the requirements were elicited. The architecture provides information concerning the location of the appropriate *expertise* to solve a problem posed by a user making use of the artifacts and experts available in the organization.**

Keywords-knowledge; expertise; agents; software development.

I. INTRODUCTION

Today, organizations have a special interest in treating knowledge as an organizational resource. This type of resource represents a change regarding how to manage information that generates different processes inside the organizations. Using this information, organizations promote the exchange of knowledge among its members, so that through the current knowledge, it will increase and improve their work practices in order to create organizational knowledge. For this, organizations need to promote internal support among members to generate and transfer knowledge that will later help with better decision making [1].

According to [2], knowledge can be found in persons (individual or group) and artifacts (e.g., business practices and daily routines, technologies, physical or digital documents repositories, such as books, manuals, videos, and so on) by which it can produce wealth, multiply the production of physical goods, and create competitive advantage [3][4]. New knowledge always stems from an individual in the organization, so that knowledge passed from individual to organizational. This means, an organization cannot create knowledge without an individual initiative and the interaction that occurs within teams. Knowledge can increase or consolidate in the group through dialogue, discussion, exchange of experience and observation. So the members of an organization generate new points of view through dialogue and discussion. This dialogue can include considerable conflicts and disagreements, but it is precisely such conflict that pushes employees to question existing premises and give new meaning to their experiences [5].

In many situations, when developers encounter difficulties with an activity, they usually searches for knowledge. This is because the goal is to find the *expertise* (better quality knowledge) to solve the problem or the difficulty, demanding a greater degree of knowledge [6]. This article addresses the problem to locate the appropriate *expertise* to solve problems in an organization, particularly in software development. This is, because the software industry generates different type of artifacts during the development process (e.g., system requirements, modules, components software, manuals, etc.) and such artifacts are connected or related to the creator/s (programmer, software architect, analyst, etc.) [7]. During the activities of software development, project activities are distributed to members of the working group and they work individually, at a certain times come together to integrate their products to achieve the project deliverables or artifacts. For this, they require existing *expertise* in the organization so they turn to their colleagues to share knowledge in order to solve obstacles. So, the challenge for these organizations is to obtain adequate *expertise* in a timely manner in order to maintain the competition level of the company to win more contracts and fulfill their commitments on time with customers, and for this the organization requires its members to be effective in generating artifacts [8].

In software development, a lot of knowledge can be shared among members of development either between analysts and programmers or between programmers and testers, but much of this knowledge remains tacit and depends heavily on the interaction between members to obtain or share that knowledge. This, combined with the global trend of the software industry [9] placing the different team members working in different geographical locations, limited to communication [10] and activities coordination [11], which is reflected in misunderstandings, errors and waste of resources [12]. Therefore, the main objective of this article is to design and model an agent based system to support the localization of *expertise* in the development of software architecture, which makes use of existing knowledge and experience, so that, in this way, they can anticipate problems, innovate and generate new knowledge between the working groups and could potentially help these organizations to improve their development process. The remainder of the paper is organized as follows Section II outlines the related work, after which, in Section III, the phases of the methodology used in this work are described. Section IV explains the multi-agent architectural propposal and describes the process of *expertise* location. Finally, Section V presents our concluding remarks and the directions of our future work.

II. Related Work

The location of *expertise* in software development is a complex and little discussed issue, since there is a tacit knowledge in people. In order to acess that knowledge, software engineers generally interact with each other to find out who is the best person to answer any questions or help with a project task. For this, efforts have been made to manage the location of *expertise* by using an agent-oriented approach, as in [13], where a multi-agent architecture designed to manage information and knowledge generated during maintenance of software is presented, using Web technologies to support interaction between players using different reasoning techniques to generate new knowledge from prior information and learning from their own experience. This reasoning is based on the different cases that happened in a previous project or previously in the same project, as well as the experience documented by the stakeholders. Another proposal that works with the agent-oriented paradigm [14] presents an initial implementation of a system that works with agents that help localization *expertise* (experts) under the theme 'Java programming language', where agents are responsible for scheduling appointments for the exchange of knowledge, proactively detecting when help is needed, providing additional information during the interaction and adjusting their own mechanisms of profiles according to user feedback.

Systems that focus to a particular kind of knowledge (artifacts) have also been developed, such as BluePrint [15], which describes the design, implementation and evaluation of a web search interface integrated with the development environment Adobe Flex Builder that helps users locate code examples of previous projects using keywords (e.g., programming language, framework, class name and/ or method). SNIFF [16] works with the location of artifacts, which facilitates the search for existing libraries through a plugin for the Eclipse development environment for Java programming language. It is based on the premise that libraries are fine match documented facilitating the search for the library with the available domain Java programming. Similarly, Exemplar [17] is a tool for searching software projects of great importance for source code reuse. It uses the keywords and the words in the description of the project to infer on the needs of the user.

Finally, there are many works about locating the knowledge of the people (experts), such as QuME [18], which is a prototype of a personalized Web interface for users of online communities requiring help with Java programming language. It has a mechanism to infer the level of knowledge of a java programming language user, calculated using parameters such as the questions being asked in the forum, the response frequency, the keywords in the user profile and other aspects that help determine the level of *expertise* of person. Expertise Recommender [19] is an experts recommendation system using a general recommendation architecture based on a study of location field experience, which is adjusted according to the needs of the user and the field of the related experts. This work has led to locate the level of knowledge of people based on specific parameters and the history of knowledge that has been shared in a specialized forum.

This section shows how researchers have made many efforts to locate the *expertise* in software development environments. These studies have suggested reasoning to generate new knowledge from a knowledge base. They have also proposed software platforms to locate artifacts in development projects and systems to allow the location of experts on specific topics. However, none of the works we studied has integrated artifacts and experts. These works are limited to collecting the knowledge for use at a particular time, without sharing it, so that no one else can access it later or find out who is the supplier. The most cited papers collect the individual *expertise* of users but do not integrate and share it to make it accessible to all developers in an organization. This is a key element, since software development experience is an important factor for on-time deliveries, as highlighted in [2]. It is important to support the reuse of organizational knowledge [6].

III. Knowledge Flow Identification

To identify barriers to the flow of knowledge and interaction in software development activities KoFI Methodology (Knowledge Flow Identification) [20][21] was used. This methodology consists of four phases. Phase 1 consists in the identification of different sources that generated or stored knowledge; Phase 2 identifies the types of knowledge used and generated in the main processes of the organization, while Phase 3 identifies how knowledge flows within the organization. Finally, Phase 4 consists in identifying the main problems that hinder the flow of this knowledge.

This methodology identified the sources of knowledge, knowledge entities that are used in the activities of these organizations, how knowledge is distributed and the problems that arise during the activities of the members in order to solve their doubts or complete their deliverables. This methodology helps the elicitation of the requirements to support the *expertise* location in the software development.

A. Phase 1: Knowledge Sources

The sources of knowledge that exist in software development organizations were validated through a focus group. This focus group was aimed at knowing how workers of software development perform *expertise* search and how they manage their knowledge, in order to validate whether the sources of knowledge that are listed in the software engineering literature are consistent with the daily developer activity. The focus group was held on the premises of the software company UXLAB [22] and lasted 90 minutes. The group involved 11 members of the organization, 7 developers and 4 designers. The focus group subjects were asked how they obtain knowledge individually, where they stored it, how they transfer it and how they infer the level of *expertise* of a colleague. The result of this phase is summarized in Table I.

TABLE I. SOURCE KNOWLEDGE

Sources	Description
Artifacts	*Bookmarks:* When developers find a place of interest or it helped them to solve a problem, they usually save the address in the browser.
	Projects: Developers usually save previous projects source code for reuse in some cases.
	Manuals: In some cases the acquired knowledge is stored as a user manual.
	Official documentation: In some cases, developers often visit the official site of some technology to obtain information about it.
Persons	Developers often seek experts, people with some degree of knowledge of a specific topic or area that could be particularly useful to solve some problems presented by development activities, so it involves all people within the organization represent a source of knowledge.

B. Phase 2: Knowledge Topics

In order to categorize and organize knowledge identified in software development organizations a plot showing the three scenarios in which the *expertise* is involved (see Table II) was performed.

The scenario of *Individual Knowledge* is characterized by a *knowledge necessity,* this consists of a developer looking for information on forums, official documentation and in his own bookmarks. Once the developer finds useful information, the *transition* (*tacit to explicit*) starts. This happens when the developer stores this information.

The scenario *Knowledge Exchange* is characterized by the way knowledge is transfered, which can be obtained through training *directly* with an expert (e.g., advice and/or training) or *indirectly* performed by using some explicit action recommended by experts (e.g., manuals, specialized forums, etc.).

Finally, the scenario *Expert Search* relates to the identification of the experience and knowledge that colleagues have about a particular query to acquire new knowledge or solve an obstacle concerning any work activity. Once a developer finds an expert, they exchange *Knowledge Exchange*.

TABLE II. KNOWLEDGE REPRESENTATION

Scenario	Characteristics	Representation
Individual Knowledge	Knowledge necessity	Forums
		Official Documentation
		Video Tutorials
		Bookmarks
		Projects
	Transition (tacit to explicit)	Format
		Author
		Thema
		Folder name
		Notes
		Language
		Source
Knowledge Exchange	Direct	Training courses
		Advisery
	Indirect	Keywords
		Bookmarks
		Web sites
		Manuals
Expert Search	Expertise Level	Experience
		Work area
		Availability
		Programming languages
		Recommendations
		Contributions

C. Phase 3: Knowledge Flow

The flow of knowledge to locate *expertise* starts with a developer that needs to acquire new knowledge to solve any difficulties in an activity. Then he/she choose between doing a search in artifacts or search for an expert.

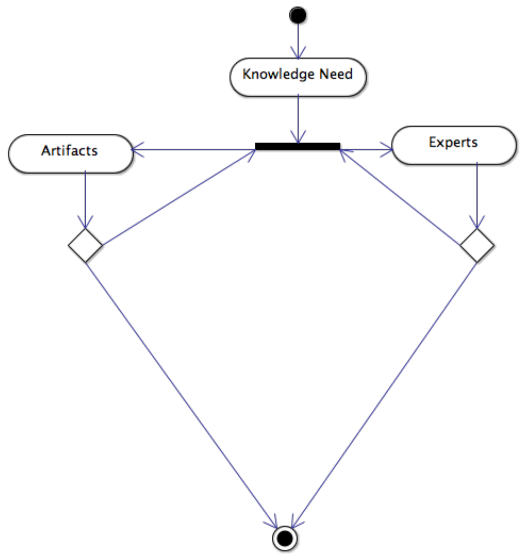

Figure 1. Knowledge Flow Activity Diagram.

In the case of choosing a search of artifacts, searches should be performed in all available artifacts (pages, manuals, videos, reused code). When an artifact is found it is checked whether this helped achieve the objective or still need to find more artifacts. If not necessary to search for more artifacts then the objective was met, otherwise, the search for more artifacts continues. An artifact may suggest to seek expert help. In the case of choosing an expert search, start looking for experts with the degree of knowledge to solve the difficulty presented in the activity. Once an expert is found, one needs to check their availability to start an interaction with the expert. Later, one must check if expert consultation was sufficient or if there is a need to consult another expert. It may also happen that an expert suggests the consultation of an artifact. If the objective was fulfilled, the process ends (see Figure 1).

D. *Phase 4: Problems in the Knowledge Flow*

With the information obtained from the focus group, it was possible to identify the problems that prevent the distribution of knowledge in the activities of support software localization *expertise*. These are:

1) *Administration of the artifacts (individual or group):* In some cases, the knowledge supplier is known, but one does not have access to its artifacts (e.g., blogs, manuals, reused code).
2) *Management Experts:* Sometimes it is difficult to find the person with the appropriate level of *expertise* to consult if there is any doubt on how to solve a problem or to perform an activity.
3) *Availability of the Experts:* In some cases, it is not known if the *expertise* or expert is available to the person who needs them.
4) *Timely resolution of difficulties*: In some cases, a lot of time is wasted in finding *expertise* that does not have the knowledge needed to solve an issue.

IV. SUPPORTING EXPERTISE LOCATION

With the information obtained from KoFI methodology, it was possible to elicit the system requirements for the *expertise* location (see Table III). With these requirements, a model of a multi-agent architecture is needed. This multi-agent system is composed of virtual modules each dedicated to managing artifacts and experts within the organization.

TABLE III. EXPERTISE LOCATION REQUIREMENTS

Requirements
R1. The system will have an interface where an information query can be performed and also new knowledge can be registered.
R2. The system is responsible for capturing new knowledge repositories and will perform searches of users.
R3. The system is responsible for making decisions to find the best resources for the user according to their needs.
R4. The system is responsible for making the calculation of potential experts that have the knowledge and availability to provide knowledge to the user.

The purpose of ExLoc (Expertise Location) system is to provide appropriate resources to the users needs, collect all the knowledge that is available within the organization, provide contact information with experts who can help solve some problems. There are 4 basic ExLoc agents: User Agent (UA), Central Search Agent (CSA) and Fuzzy Expert Agent (FEA). Working together for *expertise* location, either to capture or search (see Figure 2).

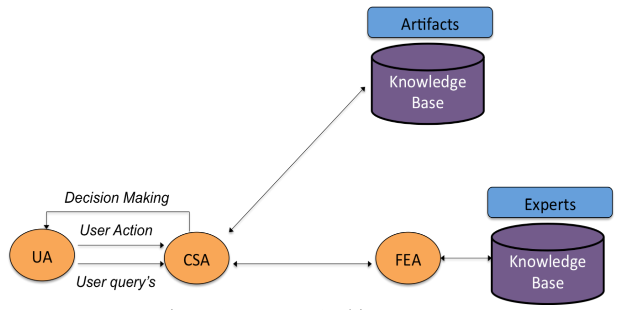

Figure 2. ExLoc Architecture.

The CSA Agent receives the search parameters sent by the UA Agent. For example, when a user needs to know *Web backend* programming, in *PHP* language to make a *database connection* in *MYSQL*, this parameters trigger a set of rules to identify artifacts or experts who have knowledge related to the search. Later the results of this process are sent to the UA agent presented as a list of reources to the user. The CSA Agent sends a series of actions that the user can perform to satisfy their need for knowledge. At the end of this process the systems ask the user if such knowledge was useful for him, using a ranking score to mark the usefulness of the resource in the case of those who have been helpful to the user. The results of this questions are used to update the knowledge repositories.

A. *User Agent (UA)*

The main objective of User Agent (UA) is to provide the user an interface that allows to search *expertise*, as well as to capture new knowledge.

Task (*R1*)

- Receive data capture of the new knowledge
- Perform Search of *expertise*
- Send the information to the Central Search Agent (CSA)
- Update repositories

There is an UA which interacts with each user allowing customization of searches through the interest or the history of users. Each UA communicates with the CSA, so that, it sends alerts or actions performed by the user and also receives the information obtained as a result of these actions and later shows it to the user.

B. *Central Search Agent (CSA)*

The main objective of CSA is to act as an expert in *expertise* location strategies in software development.

Task (*R3*)

- Decision making
- Run the search or capture
- Distribute knowledge

There is a CSA around the multi-agent system responsible for processing all actions and inferring what is the best option according to user needs.

The CSA Agent receives as input the user query. According to the need of the user, the CSA Agent creates a strategy to search in the knowledge base (KB) of artifacts and experts for the right tools to support the user. The CSA has an *inference engine* that interacts with the artifacts and experts KB. Next we describe the inference engine.

The knowledge base (KB) contains all the knowledge of the artifacts and the experts respectively. The system starts with the knowledge provided by the registered users. Through the use of such knowledge, this will increment. The semantic network of Figure 3 represents the knowledge. The expert system can search the *expertise* in any of two knowledge repositories: the artifacts contain knowledge from a number of tutorials, papers, web sites and forums. Such knowledge is classified as in the previous example where the platform was *web*, type *backend,* language *PHP* and the subject was a database *connection* in *MYSQL*. Then, the classification of the expert knowledge includes the *project* where the expert is currently working, *experience* of topics the expert has been working on (e.g., Android Projects, Web pages etc.), the *schedule* of the user to know if the expert is available and the *personal data*.

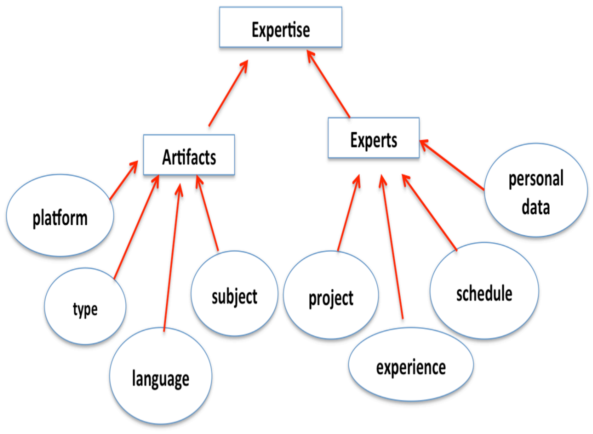

Figure 3. Semantic Network of the Knowledge Representation

The inference engine is able to explore through all the knowledge from artifacts and experts.

The inference engine begins the decision making by exploring the user query input and comparing it with the KB of artifacts and with the available experts. Figure 4 presents the inference engine implicit in the CSA agent. The user introduces the input query to the CSA. The input query is processed by the inference engine and returns a suggestion to the CSA. The inference engine uses the *expertise* stored in the knowledge base to support in the decision making process.

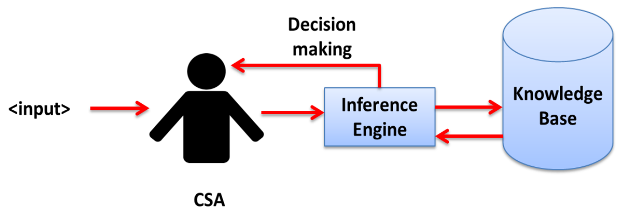

Figure 4. Inference Engine

C. *Fuzzy Expert Agent (FEA)*

FEA has the main objective to make the calculation of available experts who could help the user based on the requirements of the problem.

Task (*R4*)

- Finding the right persons according the user needs

There is a FEA responsible for calculations of the experts using the repository with information from members of the company. Later, candidates found are sent to CSA.

V. CONCLUSION & FUTURE WORK

This article addresses the issue of *expertise* location in the software development with a multi-agent system that makes use of the experts and artifacts available in the organization. Because of that, both artifacts and experts play an important role, since, in some cases, there is not enough knowledge stored in the system, but some expert may be able to help and an artifact can lead to such an expert. The requirements of the system were elicited with the information obtained from the KoFI methodology. This methodology was used to know the sources that are located in the software development software, and the topics or the way they store and transfer. Then, with this information, the knowledge flow was built, to identify the problems that the developer currently faces.

As future work, we intend to work in the semantic network presented as the basis of a language through wich the agents will comunicate. Also, we intend to extend the functionality of CSA and FEA to interact with a wider range of knowledge and perform the validation of ExLoc in the work context of software development.

REFERENCES

[1] E. Pasher and T. Ronen, The Complete Guide to Knowledge Management: a Strategic Plan to Leverage Your Company's Intellectual Capital, John Wiley & Sons, 2011.

[2] I. Becerra-Fernandez and R. Sabherwal, Knowledge management: systems and processes, ME Sharpe, 2010.

[3] E. Fragouli, “Intellectual Capital & Organizational Advantage: an economic approach to its valuation and measurement,” Business and Management, vol. 7, no. 1, 2015, pp. 36-57.

[4] N. Bontis, "Assessing knowledge assets: a review of the models used to measure intellectual capital," International Journal of Management reviews, vol. 3, no. 1, 2001, pp. 41-60.

[5] I. Nonaka and H. Takeuchi, The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, 1995.

[6] K. A. Ericsson, M. J. Prietula and E. T. Cokely, "The Making of an Expert," Harvard Business Review, vol. 85, no. 7/8, 2007, pp. 114.

[7] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7/e, Mc Graw-Hill, 2009.

[8] I. Rus and M. Lindvall, "Guest editors' introduction: Knowledge Management in Software Engineering," IEEE software, vol. 19, no. 3, 2002, pp. 26-38.

[9] D. Damian and D. Moitra, "Guest Editors' Introduction: Global Software Development: How Far Have We Come?," Software, IEEE, vol. 23, no. 5, 2006, pp. 17-19.

[10] R. E. Jensen, "Communication Breakdowns in Global Software Development Teams: Is Knowledge Creation the Answer?," Proc. 17th ACM International Conference on Supporting Group Work, ACM, 2012, pp. 289-290.

[11] J. D. Herbsleb and D. Moitra, "Global Software Development," Software, IEEE, vol. 18, no. 2, 2001, pp. 16-20.

[12] H. Van Vliet, "Knowledge Sharing in Software Development," Proc. 10th International Conference on Quality Software (QSIC), IEEE, 2010, pp. 2-2.

[13] O. M. Rodríguez, A. Vizcaino, A. I. Martínez, "Using a Multi-agent Architecture to Manage Knowledge in the Software Maintenance Process," Knowledge-Based Intelligent Information and Engineering Systems: 8th International Conference, KES 2004, Wellington, New Zealand, September 20-25, 2004, Proc., Part I, M. G. Negoita, et al., eds., Springer Berlin Heidelberg, 2004, pp. 1181-1188.

[14] A. Vivacqua, "Agents for Expertise Location," Proc. AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace, 1999, pp. 9-13.

[15] J. Brandt, M. Dontcheva, M. Weskamp and S. R. Klemmer, "Example-Centric Programming: Integrating Web Search into the Development Environment," Proc. SIGCHI Conference on Human Factors in Computing Systems, ACM, 2010, pp. 513-522.

[16] S. Chatterjee, S. Juvekar and K. Sen, "Sniff: A Search Engine for Java Using Free-Form Queries," Fundamental Approaches to Software Engineering, Springer, 2009, pp. 385-400.

[17] M. Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk and C. Cumby, "A Search Engine For Finding Highly Relevant Applications," Proc. Software Engineering, 2010 ACM/IEEE 32nd International Conference on, IEEE, 2010, pp. 475-484.

[18] J. Zhang, M. S. Ackerman, L. Adamic, K. K. Nam, "QuME: A Mechanism to Support Expertise Finding in Online Help-seeking Communities," Proc. 20th annual ACM Symposium on User Interface Software and Technology, ACM, 2007, pp. 111-114.

[19] D.W. McDonald and M.S. Ackerman, "Expertise Recommender: A Flexible Recommendation System and Architecture," Proc. ACM Conference on Computer Supported Cooperative Work, 2000, pp. 231-240.

[20] O. M. Rodríguez-Elias, A. Vizcaíno, A. I. Martínez-García, J. Favela and M. Piattini, "Knowledge Flow Identification," Encyclopedia of Information Science and Technology, 2009, pp. 2337-2342.

[21] O. M. Rodríguez-Elias, A. Vizcaíno, A. I. Martínez-García, J. Favela and M. Piattini, "Studying Knowledge Flows in Software Process," Software Engineering and Development, Nova Publishers, 2009, pp. 37-68.

[22] UXLAB, 2015; http://www.uxlab.mx/