An Extensible Service for Experts Recommendation on Distributed SoftwareDevelopment Projects

José Teodoro da Silva, Marco Aurélio Gerosa
Departamento de Ciência da Computação
Universidade de São Paulo (USP)
São Paulo, Brazil
{jteodoro, gerosa}@ime.usp.br

Igor Wiese, Reginaldo Ré, Igor Steinmacher
Coordenação de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão, Brazil
{igor,reginaldo,igorfs}@utfpr.edu.br

***Abstract*—a common challenge in Distributed software development (DSD) is the identification and ranking of experts that can provide help to team members on their tasks. This paper presents a survey conducted with distributed software developers to identify a set of requirements to improve collaboration among team members by providing experts location features. We submit an architecture that was proposed based on the survey results. Therefore, the developed architecture enables the addition of new mining and ranking methods in one search engine, also enabling the use of different ways to present the ranking.**

Keywords-Distributed Software Development; expert recommendation; architecture;

I. Introduction

Software development industry has been gaining competitive advantage in terms of cost and quality using qualified professionals distributed from all over the world [1]. This approach, called Distributed Software Development (DSD), is based on geographically dispersed teams working collaboratively in a project. Besides its advantages, DSD brings new challenges such as contextual, cultural, geographical, and temporal differences among developers [2].

During software development, team members typically share and exchange knowledge about their work by interacting constantly with others developers [3]. Traditional face-to-face communication becomes less common and interaction among members requires the use of technology to support communication [4][5] to reduce the problems caused by geographical distance [6][7].

Information sharing, which is a critical factor during software development activities [6], becomes more difficult in distributed settings because the access to remote team members is not trivial, due to temporal and physical distance. A developer often needs to identify who is the person he/she needs to talk to in order to solve an issue [8][6]. The time spent to perform this search can be improved if a tool to identify experts is provided.

The problem of searching for experts is a challenge that distributed teams face every day. Many recent studies have being conducted and tools are being proposed to support experts location on the software development domain [9][10][11][12]. However, each of these tools presents a specific method to identify the experts and just one way to present the list of experts. These tools are not extensible enough, as they cannot accommodate changes or new requirements with few or no changes in the application [13]. Thus, their use is limited to a specific situation or need. These limitations motivated an extensible architecture.

We conducted a survey with distributed teams' members to check their needs regarding expertise search and we designed a service architecture to fill these requirements. On survey, we asked participants to grade 10 different information according to the importance given to it when working on a specific software artifact. We also asked them to reveal how they find experts when they need some support on their work. The preliminary results, presented in this paper (Section 3), show that team members want to be aware of experts that can support them on their tasks; that they usually spent their time manually looking for these experts; and that they use different sources and methods to perform this manual search.

This work presents a new extensible service architecture that enables the addition of new mechanisms and techniques without architecture or clients code changes. The rest of this paper is organized as follows. Section 2 contains the related works. Section 3 presents the survey results. Section 4 discusses the service architecture. Section 5 concludes the paper and describes future work.

II. Related Work

There are many studies and tools proposed to support communication among members of geographically distributed teams [14][10][9][15]. The tools do not focus on communication itself, but enable the identification of experts within the team, providing access to the experts' skills.

Expertise Browser (ExB) [10] is a tool that uses the concept of Experience Atoms (EAs) to represent the expertise unities gathered from SCMs and content management systems. EAs are used to generate a socio technical network (graph) involving the relationships among artifacts, people, and tasks. It is used to identify experts and trace their relationships.

SmallBlue [9] uses social networks generated from messages exchanged via emails and instant messenger to rank the experts. The analysis is performed associating names and topics extracted from the texts of messages. The experts search is made via web. A user can provide a particular topic, and the system generates an ordered list with the experts on that subject.

Codebook [11] is a tool that generates a social network from source code repositories, messages, and documentation. Links are established between activities (work items), their artifacts, and developers involved. The resulting network can be used by tools, such as the Hoozizat [11], a web search portal to search for specific experts on features, APIs, products, or systems.

Presley [12] is a tool that also aims to facilitate access to experts on a given software project. The ranking of the experts is made through the SCM change history and records of messages. Presley provides the expertise ranking by means of a synchronous communication messenger, enabling team members to identify the experts and to communicate with them.

Emergent Expertise Locator (EEL) [16] is another instant messenger that displays experts on a given subject. It uses the change history of source code to rank the experts of a given artifact. To sort the experts, the tool makes use of the coordination matrix proposed by [3]. EEL enables the exchange of synchronous and asynchronous messages among team members.

All the studies aforementioned aim to improve collaboration among team members by providing experts location features. Each one has its own mechanism for identification of experts and data mining. Each tool was created for specific contexts and they do not provide any way to change or extend the ranking mechanisms neither information sources. Then, if distributed software developers want to use these ways to support their work they should use a set with many tools to search for experts on different contexts with different needs.

In addition, the clients that present the list of experts ranked are also static, defined specifically according to the project needs. Some of them show the information via web browsers, other via Instant Messengers clients, and this cannot be easily changed or redesigned to attend to other needs. These tools were not created aiming their extension and do not foresee that future mechanisms could be added. Therefore, developers should create a completely new tool to new requirements that these tools do not fulfill.

III. SURVEY CONDUCTED

We conducted a survey via web from March to April 2011 to identify what information distributed software developers consider important in DSD settings and how themsearch for people who can provide support on their tasks. This survey aimed to identify a set of requirements to improve collaboration among team members by providing experts location features. We send it to members of three companies that work with globally distributed teams andwe also published the survey in specific forums on the internet. The survey results were answers from 30 respondents of 5 different countries, all of them members of distributed software development projects. Individuals who participated in the survey had development experience between 3 and 30 years (in average, 11.9) and between 1 and 10 years (4.3 in average) working in distributed development.

We divided the survey into two sections: the first related to the respondents profile; and the second related to distributed development. In the second section, we asked what information distributed software developers consider important while developing and how they identify or locate experts on a given feature when they need support.

Regarding the information that people consider important, we presented 10 different types of information to the respondents. They were able to assign weights from 0 to 5 (0 meaning not important at all; and 5 meaning essential). The three types of information considered the most important were, in order:

- Which developer is the expert on this artifact (Average 4.07)
- From which artifacts does this artifact depends on (Average 4.03)
- Which other artifacts depend on this artifact (Average 3.97)

The survey also showed that when it is necessary to find some expert, 62% of respondents manually look for information on Software Configuration Management (SCM) systems, documentation, forums, or email threads. This indicates the range of information that can be used to locate experts.

In addition to the question regarding information that people consider important, developers where asked about the way they search for experts to help on their tasks. The answers received vary and include manually search on forum threads; research on current documentation; ask close colleagues if they know who can help; ask the responsible; and manually search for top or recent committers on SCM.Some developers also answered saying that they use different sources of information (e.g., SCMs and forums; SCM and social network) to find the best person to support them.

Moreover, they use different techniques to find the experts. The developers that use SCM to search go for the top committers, or to the last committers. Developers that use forums or mail threads find people that answer a great number of questions related to the problem to be solved, and some raise the question and wait for an answer to start a conversation. Other developers reported that they use their personal contacts to find the experts and other ones reported that choose more than a single method and one or more different sources to find the expert.

We have also asked developers how comfortable they feel to start a synchronous conversation with an expert they do not know. Some developers answered they have no problem in starting a conversation and feel comfortable with it and some developers told that they do not feel comfortable if the expert was never introduced to them. Therefore, we cannot assume that just one way to show ranking expert results.

Individuals who participated in the survey are from five different countries (India, Brazil, USA, Canada, and Italy), and most of them have colleagues working in another country. So, cultural differences must be considered. Team members are also in different time zones (sometimes there is no overlap of their work times).

We listed the following requirements to create a tool to find experts analyzing the related answers:

R1- An expert locator should provide a single way to access several information sources about experts associated with a project because it depends on the user needs, what is available on the project, and the subject;

R2- Just like data sources, the tool should support several methods because developers can choose one or more algorithms to rank the experts;

R3- Some developers feel more comfortable than others do when they have to start an interaction with an expert they do not know. Thus, an expert finder needs different ways to show results;

R4- Members of global teams can be in different countries, resulting in different cultures and time zones, so the client that displays the ranking of experts must be adapted to it and should enables different ways to interact with other member.

We verified that related tools do not fill these requirements. Thus, in this work, we aim to fulfill these requirements by providing an extensible service architecture that enables the use of different ways to identify experts and different clients to display the results with a web service infrastructure

IV. Service Architecture

As observed in Section 2, developers usually search for experts manually in SCM systems, forums, documents, etc. This search can consume a significant amount of their time. We propose a service framework to identify, and rank team members, so that software professionals do not spend their time manually looking for experts [10]. The architecture enables the definition of different methods of ranking. To create a platform-independent architecture, we designed it as a service. Service-oriented architecture helps to achieve extensibility and scalability of systems [17].Therefore, it is possible to create third-party clients to consume the service proposed. This enables one to create a client that presents the results in different ways to best suit user needs, and enables existing clients to use any available ranking mechanisms without having to change clients' code.

The clients and the service interface communicate via XML request and response messages, enabling any third-party client to send a request and consume the response of the service. Thus, the clients can display experts according to their needs, based on the results generated by the service. Hence, they can use the same information to request and display the experts in web portals, just implementing an XML parser for the new client.

The service architecture proposed is divided into three layers: communication, mining, and classification (Figure 1). A detailed description of the layers and how they address the requirements raised in Section 3 follows. The communication layer is an interface that provides means for the information exchange between clients and the ranking and mining services. It receives the information needed to perform the ranking, such as the ranking mechanism to be used and the subject to which the expert is required. This layer enables the users to consume the same service in different clients.

This feature addresses the requirement of providing different ways to present the experts information according to the geographical location of the users, considering cultural and temporal differences (R4). A user can choose the best way to find the experts and to start an interaction that best fits with his/her location and culture. Besides cultural and temporal differences, some respondents assumed they do not feel comfortable to have a first contact with an unknown person (R3). In this case, a synchronous tool is not welcome. A client that displays the experts and their social networks would be a good solution. Presenting the contacts that the developer has in common with the expert can help by showing who can introduce them to start a conversation.

In addition, different clients would be useful as team members use different tools to perform their tasks and have different roles. Each member can have their own client embedded within their tools. In case they have different roles, they would like to use different ranking mechanisms that can be set on the client and sent via communication layer to the layers below, generating a different list.

The ranking layer contains the ranking algorithm implementations, presenting an ordered list of experts for the subject. This layer can contain various implementations of different techniques and use the communication layer to Figure 1.The survey results showed that current expert searches are made manually following different algorithmsand, using data from a one or more different sources (R2). This layer enables the use of different methods of ranking the experts and different algorithms with data coming from one source, or with data coming from multiple sources can be generate the ranked list.

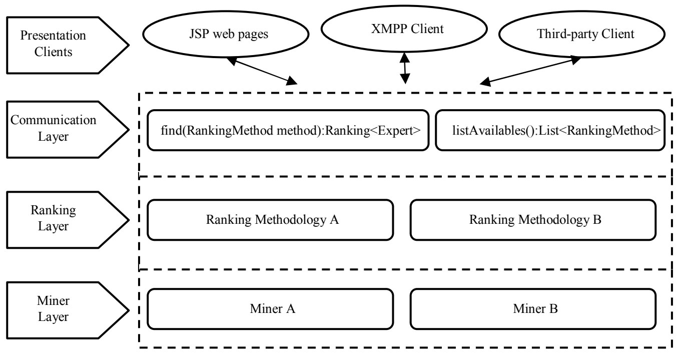

Figure 1. Three layer service architecture and some clients

The survey suggests that there is no common sense about the best source to find experts. The mining layer contains the mining techniques that enable to use many information sources to rank the experts (R1). This layer gets the data from different sources to rank the team members expertise. The ranking mechanisms implemented can provide one or more mining module to extract the data from required sources or use modules already present in the application.

Besides the service-oriented structure, we have used paradigms and techniques that promote the extensibility and scalability of ranking methodologies. We chose an architecture based on design patterns and interfaces to achieve an extensible service structure, In this way, on the server side the ranking mechanisms and the mining techniques are interfaces in the architecture. These interfaces are the service hotspots where third-party can add new methodologies and techniques. On the other hand, clients can use these extensions without any code change.

V. CONCLUSION

Developers need accurate information to perform their tasks, promoting time and cost reduction and avoiding conflicts and rework. In addition, team members to be shared information about the project need, and the difficulty in transferring this knowledge can affect team performance. In this context, team members use tools to facilitate information exchange and promote expertise identification. This work reported a survey conducted with developers of distributed teamsto identify a set of requirements to improve collaboration among team members, and presented a service infrastructure that facilitates the expertise search and the information exchange. The analysis of the answers of the survey resulted in four requirements to be fulfilled when designing a solution to support experts' location (presented on Section 3). The service architecture presented enables the addition of new clients, ranking, and miners without major efforts.

The architecture is a web service and is divided into communication, mining, and ranking layers. The communication layer is an interface that receives requests and redirects the answers. It enables the creation of different types of clients to consume the service, meeting the requirements R3 and R4 (extracted from the survey), and enables the creation of clients embedded into different tools and useful for different roles. The ranking layer enables the definition of different algorithms and methods for experts' identification and ranking, fulfilling the requirement R2. Finally, the mining layer enables the data acquisition from different sources (mails, SCMs, forums, documents) addressing the requirement R1.

Up to now, the mining strategy implemented support the extraction of information from Subversion (SVN) source code repository. The next steps on this research are the service extension by third-party to evaluate the architecture extensibility and an extension to enable us to overcome the other two information considered the important ones according to the survey results presented in Section 3.

REFERENCES

[1] M. Robinson and R. Kalakota, Offshore Outsourcing: Business Models, ROI and Best Practices, Alpharetta, GA: Mivar Press, 2004.

[2] M. Ivcek and T. Galinac, "Aspects of Quality Assurance in Global Software Development Organization," in Proceedings of the International Conference on Telecommunications and Information of the 31st International Convention MIPRO 2008, Opatija, Hrvatska, June 26 to 30, 2008, 2008.

[3] M. Cataldo and J. D. Herbsleb, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity," in Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, New York, 2008.

[4] M. Jimenez, M. Piattini and A. Vizcaíno, "Challenges and Improvements in Distributed Software Development: A Systematic Review," Advances in Software Engineering, vol. 2009 , pp. 1-14, January 2009.

[5] I. Steinmacher, A. P. Chaves and M. A. Gerosa, "Awareness support in global software development: a systematic review based on the 3C collaboration model," in 16th International Conference Collaboration and Technology, Maastricht, The Netherlands, September 20 to 23, 2010, 2010.

[6] C. C. Trindade, A. K. O. Moraes and S. L. Meira, "Comunicação em equipes distribuídas de desenvolvimento de Software: Revisão sistemática," in THE EXPERIMENTAL SOFTWARE ENGINEERING LATIN AMERICAN WORKSHOP, Salvador, Brasil, 2008.

[7] K. Ehrlich and K. Chang, "Leveraging expertise in global software teams: Going outside boundaries," in Proceedings of the IEEE international conference on Global Software Engineering, Florianopolis, Brazil, October 16 to 19, 2006, 2006.

[8] A. Moraes, E. Silva, C. d. Trindade, Y. Barbosa and S. Meira, "Recommending experts using communication history," in Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering, New York, 2010.

[9] N. S. Shami, K. Ehrlich and D. R. Millen, "Pick me!: link selection in expertise search results," in Proceedings of the 2008 Conference on Human Factors in Computing Systems, New York, 2008.

[10] A. Mockus and J. D. Herbsleb, "Expertise browser: a quantitative approach to identifying expertise," in Proceedings of the 22rd International Conference on Software Engineering, Orlando, Florida, May 19 to 25, 2002, 2002.

[11] A. Begel, Y. P. Khoo and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, Cape Town, South Africa, May 2 to 8, 2010, 2010.

[12] C. Trindade, Y. Barbosa, A. Moraes, J. de Albuquerque and S. de Lemos Meira, "An Expert Recommender System to Distributed Software Development: Requirements, Project and Preliminary Results," in Proceeding of 2009 Simposio Brasileiro de Sistemas Colaborativos (SBSC), 2010.

[13] B. Johnson, C. Johnson, W. W. Woolfolk and R. Miller, Flexible Software Design: Systems Development for Changing Requirements, Auerbach Publications, 2005.

[14] I. Kwan, D. Damian and M.-A. Storey, "Visualizing a Requirements-centred Social Network to Maintain Awareness Within Development Teams," in Proceedings of the 1st international workshop on Requirements Engineering Visualization, Washington, 2006.

[15] K. Dullemond, B. v. Gameren and R. v. Solingen, "Virtual Open Conversation Spaces: Towards Improved Awareness in a GSE Setting," in Proceedings of the 5th IEEE International Conference on Global Software Engineering, Princeton, USA, August 23 to 26, 2010, 2010.

[16] S. Minto and G. C. Murphy, "Recommending Emergent Teams," in Proceedings of Fourth International Workshop on Mining Software Repositories (ICSE Workshop), Minneapolis, USA, May 19 to 20, 2007, 2007.

[17] N. M. Josuttis, SOA in Practice, O'Reilly Media, 2007.