GREGORIA ROMERO GRANDE

De:	GREGORIA ROMERO GRANDE
Enviado el:	jueves, 10 de septiembre de 2009 9:37
Para:	Aurora Vizcaino Barcelo
Asunto:	RV: un favor
Datos adjuntos:	CSCWD09-3485-CWS_Rpalacio_etal[FinalPaper].pdf; CSIE09_942_PID796197_rpalacio_etal.pdf

Aurora,
Necesito también las paginas de portada y contraportada, la del Copywrite y el índice, puedes localizarlas por favor?.

Gracias,

Goyi Romero Grande
UCLM - ESI - Grupo ALARCOS
Pº de la Universidad, 4, 13071 Ciudad Real
Tlf.: 926 295 300 Fax: 926 295 354, ext.: 3747
e-mail: Gregoria.Romero@uclm.es

De: Aurora Vizcaino Barcelo [mailto:Aurora.Vizcaino@uclm.es]
Enviado el: martes, 01 de septiembre de 2009 14:46
Para: GREGORIA ROMERO GRANDE
Asunto: Re: un favor

Goyi, te mando dos artículos con sus referencias para añadir a la Alarnet. Gracias
Aurora

> Ramón R. Palacio, Alberto L. Morán, Víctor M. González, Aurora Vizcaíno, /"Providing Support for Starting Collaboration in Distributed Software Development: A Multi-agent Approach,"/ csie, vol. 7, pp.397-401, 2009 WRI World Congress on Computer Science and Information Engineering, 2009. Press ISBN: 978-0-7695-3507-4. Los Angeles, USA, March 31 - April 2, 2009.
>
>
> Ramon R. Palacio, Alberto L. Moran, Victor M. Gonzalez, Aurora Vizcaino, "/Col//laborative Working Spheres as support for starting collaboration in distributed software development/," cscwd, pp.636-641, 2009 13th International Conference on Computer Supported Cooperative Work in Design, 2009. Press ISBN: 978-1-4244-3534-0. Santiago, Chile April 22-April 24

Providing Support for Starting Collaboration in Distributed Software Development: A Multi-Agent Approach

Ramón R. Palacio[1], Alberto L. Morán[2], Víctor M. González[3], Aurora Vizcaíno[4]
[1] *Facultad de Ingeniería Ensenada, UABC*
[2] *Facultad de Ciencias, UABC*
[3] *Manchester Business School, University of Manchester*
[4] *Alarcos Research Group, University of Castilla-La Mancha*
{rpalacio, alberto_moran}@uabc.mx, vmgonz@manchester.ac.uk, Aurora.Vizcaino@uclm.es

Abstract

Developing a system to provide support for distributed software developers (DSD) to get into collaboration is very complicated. On the first hand, it is necessary to know the characteristics of their daily work activities. On the other hand, technical aspects must be considered, such as obtaining information on the context and on the data flow of their activities. This requires information from the individual and group work environment. Therefore, we propose a model to help in the development of this type of system. Firstly, our work aims at modeling the information flow of DSD workers based on a literature survey and on our own experience. Secondly, we describe a projected implementation scenario based on the multi-agent proposed system.

1. Introduction

The organizations that are dedicated to software development are facing an emerging paradigm shift towards the distribution of processes and development teams. This change is due, among other things, to the desire to exploit the broader working day schedules, to benefit from the distribution of resources, lower costs and be demographically closer to the target consumer. However, there are also negative aspects such as increased risk of problems with communication [1] [8].

This new development paradigm poses several challenges for software engineering. Among others: the need for new techniques or extensions of existing ones, to support the inclusion of third-party services; new processes, mechanisms and tools to deal with the fact that the development teams (e.g. requirement analysis, design, management, etc.) are geographically distributed. This has resulted in that some organizations have had to alter the way they conduct their processes. They were used to traditional software development practices, and required to gather in one place all those involved in these processes such as customers, users, developers, testers, project managers, etc.

This new development paradigm that allows that those involved in the process be distributed in remote sites, is known as Distributed Software Development (DSD) [1]. There are a number of characteristics that define scenarios for DSD. One of these is the distance between individual members or teams, which can vary from a few meters (when the teams working on separate but adjacent buildings) up to tens, hundreds or thousands of kilometers (when they are in different cities). A special case of DSD is where the distances are among cities from different countries, even continents (global software development or GSD) [1].

As a result, people who are not co-located search for ways to be in contact with their colleagues in an informal and rapid way (e.g. to ask for clarification or help, to get others' point of view, etc.) [16], similarly as when they are co-located. The phone and instant messaging tools are typical examples of technologies that are used to try to solve these problems. However, even when these tools are easy to use, and usually ubiquitously distributed in work environments, they can have negative results [11]. This is mainly due to their lack of mechanisms that allow for a balanced way to decide whether the time for starting an interaction is suitable both for the one making contact, and for the one being contacted. A typical example of this problem is what we called "selective availability", i.e. the ability to establish one's availability according to a criterion, such as "I am available only for people who are related to the task I am dealing with and not available for other people". This problem typically presents itself in distributed work environments, where users usually do not have their partners in sight, and therefore do not know what activity is being conducted to determine whether the time is right to start an interaction.

Thus, we propose to integrate a perspective of personnel activity management of the workers in DSD with an approach to potential collaboration awareness [14] to identify suitable and appropriate moments to start collaboration, not only for the person making contact, but also for the person who is contacted in DSD processes.

The rest of this paper is organized as follows. Section 2 describes some of the characteristics of DSD. Section 3 describes a multi-agent model proposed as support for starting collaboration in an appropriate manner. Section 4 describes a projected application scenario using the proposed multi-agent system. Related work is described in Section 5. Finally, Section 6 presents some conclusions, and some directions for future work.

2. Features of DSD

DSD, as a new work paradigm, provides the following benefits: i) software companies require highly skilled human resources and seek to meet this need by employing programmers in different cities and countries [2], ii) to be closer to the target markets and to have a shorter response time, many companies have established development groups closer to the location of their client [3], iii) virtual development groups need to be created quickly to exploit opportunities in this new market [4], iv) by working in different time zones, development groups can work continuously (24/7) in critical projects [5], and v) the reduction of costs by hiring human resources in places where labor is cheaper [6].

However, DSD also faces interesting challenges, such as: i) cultural differences can affect projects in different ways, including the effectiveness of communication and coordination, the decision-making processes of a group and the performance of a team [5] [7], ii) time differences between working groups affect in activities where there is a need for intensive collaboration between the groups involved, so the synchronous communication is difficult to establish within the normal working day [8] iii) inadequate communication hampers coordination or management due to factors such as those previously mentioned, and to the fact that processes are conducted in a distributed manner, and require people to be responsible for coordinating the various involved tasks, activities and people [3], iv) DSD workers need to share a lot of information (e.g. business model, requirements, etc.) coming from different sources (e.g. customers, suppliers, requirements and system analysts, etc.) and who are at distant sites [5], and v) the greater the distance between individuals or groups, it becomes increasingly difficult to maintain relationships of trust because of the lack of an informal and spontaneous communication [9].

It is worth mentioning that work in software development environments is characterized by the existence of a high level of communication and coordination among participants [10]. This is due to the need of achieving consensus for the group's decisions (e.g. approval of requests between customers and analysts), decisions must be made known to others accurately and expeditiously (e.g. notification and delivery of new versions of a design document), interaction is regular and frequent among team members (e.g. request and delivery of information on task progression), and work is cooperative and collaborative (e.g. pair-programming). However, dealing with these characteristics is very different depending on how the development process is performed. On the one hand, in the case of co-located development project members are at sight or easily accessible, making it possible to effortlessly see or know with a simple glance what they are doing. You can even know or intuit whether a time it is appropriate to interrupt what others are doing in order to maintain communication and coordination. On the other hand, in the case of DSD, participants are located at remote sites, so that the contextual information that is readily available in the co-located case is not accessible, making difficult the processes of communication, coordination and production.

The characteristics of the activities that DSD workers perform, as detailed in [13], led us to adopt a practical work unit that allows first to understand the management of individual developer's work activities. This unit is the Working Sphere [12]. However, the working sphere concept is limited to a focus on the individual work. In contrast, the DSD context demands a focus on the work of the team or group. Nonetheless, a focus on the individual activities of collaborators is still needed.

We therefore propose to introduce the concept of Collaborative Working Spheres (CWS), which extends the concept of Working Spheres considering the work characteristics of DSD, and the design insights previously identified [13]. Some of the motivations to create this concept came from the integration of personal activity management of individual developers with the concept of potential collaboration spaces, which allow collaborators to obtain a partial and personal view of the information related to the work units which are shared with other collaborators. Thus, a CWS allows workers to detect, identify or create opportunities for collaboration between them based on the information managed from their individual work units. In addition, this allows identifying an appropriate moment to initiate collaboration in a more informed way. Also, a CWS will allow collaborators to have a meeting point with their potential collaborators, where they are actually offered with a way to start adequate interaction and from where to begin a working meeting with the collaborators and the work units involved to consistently trigger group work (actual collaboration) [14].

Therefore, the technological solution must allow that the involved collaborators and work units display the required information (e.g. presence and status). Such information must be accessed through a monitor (technological tool) that shows information on the context of users and from the same previously shared work units. To solve this problem in the next section we propose a multi-agent model designed to facilitate communication among distributed team members. The model helps to detect when it is appropriate or less disruptive to interrupt someone, thus promoting socially appropriate communications.

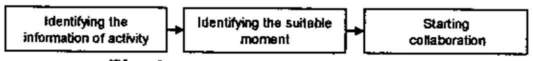
Identifying the information of activity
Identifying the suitable moment
Starting collaboration

Fig. 1. Conceptual model of the CWS

To achieve this, Figure 1 depicts a conceptual model of a CWS. It includes three main tasks: (1) Identifying the required information of the activity of those involved, (2) Identifying a suitable moment to interrupt other collaborators, and (3) Entering into collaboration if the moment is right. This requires being able to monitor the activities of the collaborators, identifying specific information from the common work unit (e.g WS) so that information on the currently shared activity could become known to the group, and based on the obtained information whether the moment is adequate for starting an interaction attempt.

3. A Multi-agent model to provide support for starting collaboration

For the technological solution proposed in Figure 1, we included the following phases: i) Monitoring: be alert to identify the work units in which collaborators are working. ii) Identification: link the files that are being manipulated by the user with the work units that were assigned to him/her. iii) Request: Perform requests for information of the current state of one or more users, based on the current state of the activity that occurs at a given moment. iv) Formalization: receive the requests from the previous phase to formalize it through the interpretation of the data that such a request brings. It must be verified whether the request is valid. v) Processing: carry out the necessary consultations to the organizational repository based on valid requests, and vi) Notification: publicize the results of the petition.

To satisfactorily address the previously mentioned phases we propose a multi-agent model (see Figure 2). Agents are involved in the following manner: i) Monitor Agent, it provides support in the monitoring phase. It is responsible for knowing the information (e.g. name, type, date, etc.) of the artifacts (e.g. files) manipulated by the user during his/her work. This requires the implementation of a proactive monitoring process to capture the information generated during the interaction between the user and the computer applications (e.g. text editors, programming languages, design applications, etc.). ii) Identifier Agent, it provides support in the identification phase. This agent is responsible for identifying whether the file that is being manipulated by the user has to do with any activity and/or task of the organization's projects. To achieve this, the implementation of a proactive search process to identify and link the information generated during the monitoring phase is required. iii) Requester Agent, it provides support in the request phase. This agent is devoted to request information by executing a process to send data to the project instance through the use of a communication channel. iv) Manager Agent, it provides support in the formalization phase. This agent will be aware of valid projects. For this, a search process on the organization's project repositories is requested to help deciding whether the petition proceeds or not. If the request proceeds the process goes to the next phase. On the contrary, the reason for request rejection should be notified to the Interface Agent. v) Project Agent, it provides support for the processing phase. It is an agent responsible for the information of a specific project. It will request a

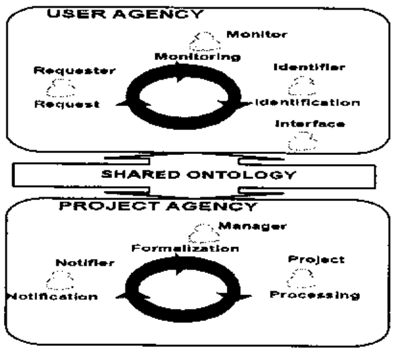

Fig. 2. Distribution of agents.

search process for a particular project of the organization and the result will be sent to the Notifier Agent. vi), Notifier Agent, it provides support for the notification phase. It is responsible for reporting the request results to the Interface Agent (mediator between the user and agents).

Agents are structured in two agencies: the User Agency and the Project Agency (see Figure 2). The User agency is responsible for providing support to requests for information by the interested user. It is composed of the Monitor, Requester and Identifier agents. It is worth mentioning that there is an assistant agent in this agency, which is called Interface Agent. On the other hand, the Project Agency aims to provide the information of the context of work based on information obtained from projects and users repositories of the organization. This agency comprises the Manager, Project and Notifier agents.

In addition to Agencies, Figure 2 shows a shared component ontology. This is necessary so that there is a consistent communication way between the agents of the different agencies.

4. Application scenario

An example of application scenario is as follows: in a DSD organization a system designer accesses a UML file through a diagramming application. This file was sent to him/her as part of an interface design task.

In the actual scenario whenever the designer has a doubt about the contents of the UML file, he/she usually tries to contact the responsible analyst by any means of communication (e.g. telephone or an instant messenging application). In this way, the designer usually interrupts the activity of the analyst.

For the projected scenario we present a scenario diagram (see Figure 3), based on the INGENIAS methodology [15].

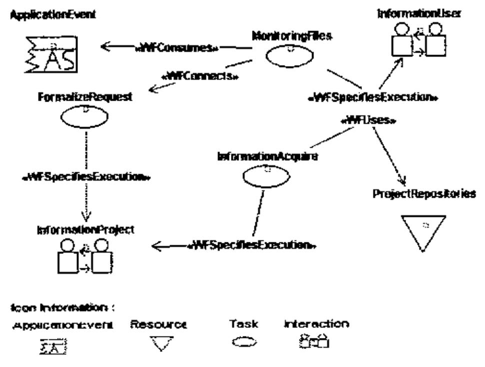

Fig. 3. Scenario Diagram

In the projected scenario, at the time when the UML file is accessed (*ApplicationEvent*), the Monitor Agent observes the event (*MonitoringFiles*) and records it in its log (e.g. file name, document type, time, date, state). Then the Identifier Agent verifies whether that file is related to a project. In this case, it verifies whether the file is associated to a task of project "X" and updates the log by marking the file as valid to make a request for information. Such a request is detected by the Requester Agent (*InformationUser*). After the interactions between agents of the User Agency, the Requester Agent makes a request to the Manager Agent, which formalizes the request validating that the Project Agent that corresponds to Project "X" is active (*FormalizeRequest*). In this case the project is active and the Manager Agent sends the information processing order to the Project Agent of the project in question (*InformationProject*). Upon receiving the order, the Project Agent searches in the repository (*ProjectRepositories*) updated information on the projects and users associated to this task (e.g. developers involved, the state of developers, state of the task, etc.). On finding the information, it is packaged and sent (*InformationAcquire*) to the Interface Agent (*InformationUser*), which is responsible for updating the user interface with the received information.

In this case, the designer obtains information regarding the file in a rapid and seamless manner through the user interface of the CWS. The presented information refers to which collaborators were involved, and in which documents related to a project they were working. In turn, this information also allows the user to interpret the current state of collaborators (e.g. busy, available, not connected, etc.).

5. Related Work

Several research works have been identified from the literature that address three main tasks to start collaboration in a proper way in DSD. These tasks include: i) Identifying the right time to enter into collaboration. Potential Collaboration [14] represents a complementary moment to Actual Collaboration. It refers to the possibility of collaboration, and as such it occurs while people are working on an individual basis, not necessarily in relation to a collaborative effort, mostly outside of a shared space representation and usually in asynchronous communication mode. Doc2U [14], offers an extended instant messaging service that provides support for potential collaboration through the presence of users, documents and specialized services. These works provide elements to create services of user, task and resource presence. ii)

Identifying information of the activity of DSD workers. Working Spheres (WS) propose a way to manage personal activities in the presence of interruptions [12]. Although it only contains information about individual activities, WS from multiple users can be grouped in a manner that they may be useful for group work. For example, they might allow to know in which "shared" activity and/or task of the WS several collaborators of the same project are working. iii) Entering into collaboration. Project-View [16] suggests modifications to the UI of a traditional instant messaging application considering three specific characteristics: awareness, user information and reminders.

However, these studies were developed for purposes other than starting collaboration in DSD.

6. Conclusions and Future Work

Currently there is a great tendency to develop software in a distributed manner, and this in order to take benefit of the advantages that this kind of development brings. However, distributed development also implies that people does not have the same opportunities for face to face interaction, which in turn introduces the possibility of coordination problems, disruptions at unwanted times, and the lack of informal communication. All of this can have an impact on the trust and quality of team member communications, and on the project results [10].

As current and future work, a prototype is being built using the following technologies as support for the multi-agent system: Web Services are being used and the SOAP protocol (Simple Object Access Protocol) is being used for the exchange of messages. An XML-based ontology is also under development, the main idea is to provide a single representation for the communication and understanding between agents.

7. Acknowledgements

This work is partially supported by FABRUM, project, Ministerio de Ciencia e Innovación, Spain (grant PPT-430000-2008-063), and by UABC under grant 0191 of the XI Convocatoria Interna de Proyectos de Investigación. The first author is supported by scholarship PROMEP/103.5/06/3244.

8. References

[1] Layman L., Williams L, Damian D. y Bures H. "Essential communication practices for Extreme Programming in a global software development team", Information and software technology , 48(9), 781-794, 2006.

[2] Ebert C. and De Neve P., "Surviving Global Software Development", IEEE Software. 18(2). 62-69, 2001.

[3] Damian, D. and Moitra D., "Guest editors' Introduction: Global Software Development: How far have we come?". IEEE Software, 23(5), 17-19, 2006.

[4] Lloyd W., Rosson M., and Arthur J. "Effectiveness of elicitation techniques in distributed requeriments engineering". Paper presented at the 10th anniversary IEEE Joint International Conference on Requeriments Engineering, RE'02, Essen, Germany, 2002.

[5] Herbsleb J. and Moitra D., "Guest editors' Introduction: Global Software Development". IEEE Software, 18(2), 16-20, 2001.

[6] Conchúir E., Holmstrom H., Agerfalk P., Fitzgerald B., "Exploring the assumed benefits of global software development", Paper presented at the IEEE International Conference on Global Software Engineering, 2006.

[7] Damián D. "Stakeholders in global requeriments engineering: Lessons learned from practice ". IEEE Software, 24(2), 21-27, 2007.

[8] Damian D. and Zowghi D., "The impact of stakeholders geographical distribution on managing requeriments in a multi-site organization", Paper presented at the IEEE Joint International Conference on Requeriments Engineering (RE'02), Los Alamitos, CA, USA, 2002.

[9] Babar M., Verner J., Nguyen P., "Establishing and maintaining trust in software outsourcing relationships: An empirical investigation", The Journal of Systems and Software, 80(9), 1438-1449, 2007.

[10] Kraut, R. E. and Streeter, L. A. "Coordination in software development". Commun. ACM 38, 3, 69-81. 1995.

[11] Czerwinski M., Horvitz E., Wilhite S. "A diary study of task switching and interruptions", Proceedings of the SIGCHI conference on Human factors in computing systems, 175-182, ACM Press. 2004.

[12] González V. and Mark G. "Constant, Constant, Multi-tasking Craziness": Managing Multiple Working Spheres. Proceedings of CHI 2004, ACM Press. 2004.

[13] Palacio R., Morán A., González V., "Soporte para la entrada en colaboración en el desarrollo distribuido de software: implicaciones de diseño", ENC-SIS 2008; Mexicali, BC, submitted for publication.

[14] Morán A., Favela J., Romero R., Natsu H., Perez C., Robles O., Martinez A, "Potential and actual collaboration support for distributed Pair-Programming", Computación y sistemas Vol. 11 No. 3 pp 211-229. ISSN 1405-5546, 2008.

[15] Pavón J. and Gómez-Sanz J. J., "Agent Oriented Software Engineering with INGENIAS." In proceedings of the Multi-Agent Systems and Applications III: CEEMAS 2003 (LNAI), Prage, Czech Republic, pp. 394-403, 2003.

[16] Fussell, S. R., Kiesler, S., Setlock, L. D., and Scupelli, P., "Effects of Instant Messaging on the management of multiple project trajectories". In Proceedings of the SIGCHI CHI '04. ACM, New York, NY, 191-198. 2004.