Proceedings of the 44th Annual Hawaii International Conference on System Sciences

Proceedings of the 44th Annual Hawaii International Conference on System Sciences

4-7 January 2011
Koloa, Kauai, Hawaii

ABSTRACTS
and
CD-ROM of Full Papers

Edited by
Ralph H. Sprague, Jr.

CPS Conference Publishing Services

Los Alamitos, California
Washington O Tokyo

IEEE computer society

Copyright © 2011 by The Institute of Electrical and Electronics Engineers, Inc.

All rights reserved.

Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries may photocopy beyond the limits of US copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or republication requests should be addressed to: IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 133, Piscataway, NJ 08855-1331.

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society, or the Institute of Electrical and Electronics Engineers, Inc.

IEEE Computer Society Order Number P4282
BMS Part Number CFP11160-PRT
ISBN 978-0-7695-4282-9
ISSN Number 1530-1605

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Lisa O'Conner
Cover art production by Joe Daigle
Printed in the United States of America by The Printing House

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

Knowledge Systems
Co-Chairs: Murray Jennex and David T. Croasdell

Knowledge Flow as Facilitator for Getting into Collaboration in Distributed Software Development

Ramón R. Palacio[1], Alberto L. Morán[2], Aurora Vizcaíno[3], Víctor M. González[4]
[1] DES Navojoa, ITSON
[2] Facultad de Ciencias, UABC
[3] Alarcos Research Group, University of Castilla-La Mancha
[4]División de Ingeniería, ITAM
rpalacio@itson.mx, alberto.moran@uabc.edu.mx, Aurora.Vizcaino@uclm.es,
victor.gonzalez@itam.mx

Abstract

The software industry is facing a paradigm shift towards distributed software development (DSD). This change creates situations from which organizations may benefit, and challenges to which they must adapt to (e.g. the absence of opportunities for informal interaction). In this paper we present the results of an analysis of the knowledge flow problems which hinder members of a DSD team to get into synchronous collaboration. We used the KoFI methodology to identify knowledge sources, topics, and existing flows, and to identify knowledge flow problems in the DSD environment. These have been used as a basis for the design and development of CWS-IM, an extended instant messenger tool which implements "Selective Availability" to facilitate "l'entreé en collaboration" of the members of a DSD team. Therefore, it is expected that the interruptions during DSD will be less disturbing as this tool helps users to decide when it is the most suitable moment to establish synchronous contact for the sender and the receiver.

1. Introduction

The software industry is currently facing a new trend towards Distributed Software Development (DSD) [1]. In essence, this tendency involves separating the different development processes and collaborators in multiple geographic locations. Geographic distribution leads to the situation of developers experiencing difficulty in interacting and coordinating their activities [2, 3]. This is caused by little or no face-to-face interaction between developers [4, 5], which is in turn caused by the distance between the developers involved in joint projects.

For example; in the case of co-located development, project members are on sight or are easily accessible, so that it is possible to see or infer what they are doing without any significant effort. It is even possible to judge whether that precise moment is appropriate to interrupt others in order to establish communication and maintain a coordination effort. In contrast, in the case of DSD, participants are located on distant sites, which means that the contextual information existing in a co-located situation is no longer present, which hinders communication, coordination and control. It is therefore essential to know the state of the activities of the person that one wishes or needs to contact. Based on this information then find an appropriate moment for both participants at which a synchronous interaction can be initiated while minimizing the negative effects of the interruption [6]. That is, good coordination requires good communication [2], and good communication requires an appropriate initial interaction.

Several empirical studies have investigated how workers are interrupted while carrying out their activities (e.g. [7]). These studies have shown that the complexity of the task, its duration, the number of interruptions and the type of task have an impact on the difficulty of returning to the interrupted task (task switching). The results of these studies thus characterize how workers behave when confronted with interruptions (interruption management). Other works study the problems that an interruption may cause (e.g. prospective memory failure). This effect increases with the number of interruptions and also when the number of tasks rises [6]. On the other hand, returning to an interrupted task is also rather difficult for people [7], as they have to remember what the last thing they were doing in relation to that interrupted task [8]. Another issue stated in [9] is that when workers are performing a task they often and ideally tend to delay their response to an interaction with the purpose of finishing the task at hand before attending the new interaction request. Some other studies also claim that there is also an increase in the interruptions caused by new technologies (e.g. email, instant messaging, mobile devices) [8]. Researchers are therefore attempting to study when these interactions

1530-1605/11 $26.00 © 2011 IEEE

are or are not irrelevant and providing solutions for increasing workplace efficiency (e.g. [9]).

It is also worth mentioning that work in software development environments is characterized by the existence of a high level of communication and coordination among participants [2, 10]. This is owing to the need to achieve consensus for group decisions (e.g. approval of requests between customers and analysts), the necessity to make decisions known to others in an accurate and efficient manner (e.g. the notification and delivery of new versions of a design document), the fact that interactions are regular and frequent among team members (e.g. request and delivery of information on task progression), and that work is essentially cooperative and collaborative (e.g. pair-programming and reviewing).

In order to facilitate interaction among developers and minimize the negative effects of work fragmentation and interruptions, it is thus necessary to obtain information about the activities undertaken by developers. It is also important to integrate this information with the organization's knowledge and to enable it to be used by team members to create new knowledge and identify appropriate opportunities for initiating synchronous collaboration.

To achieve this, it is necessary to have information elements that will help developers to be aware of their colleagues' activities.

In order to create awareness or knowledge about project collaborators' current activities in a DSD setting, it is first necessary to answer the following questions:

- *What information is needed to create awareness regarding the project collaborators' current activities?*
- *What are the sources of this information?*
- *How the information flows allow for the creation of awareness?*
- *What are the main obstacles (breakdowns) that prevent these flows in a DSD environment?*

These questions were explored and answered in a study where we used the KoFI methodology [11, 12] in order to analyze knowledge flows of a software factory in Mexico. Our aim was to obtain information for the creation of a knowledge map, which would allow us to detect the information elements, how developers interact among them during their workday, and which were the barriers that hindered these interactions and knowledge flows in a DSD setting.

In this paper we particularly report on the identified information elements, knowledge sources and main obstacles in order to create awareness about the DSD workers' activities to inform the design of CWS-IM.

The remainder of the paper is structured as follows: Section 2 describes the KoFI methodology, while Section 3 describes how this was applied to analyze knowledge flows in a software factory. Section 4 presents the design implications identified for a tool that aims to provide adequate support for initiating collaboration. Section 5 then describes the awareness information elements identified and the mechanisms used to obtain them. Section 6 shows and describes the proposed prototype interfaces of CWS-IM that resulted from considering the identified information elements and mechanisms. Finally, Section 7 discusses related work while Section 8 presents our concluding remarks and some directions of future work.

2. The KoFI methodology

The methodology used to identify barriers that hindered knowledge flow and interaction in DSD is KoFI (Knowledge Flow Identification) [11, 12]. This methodology is based on identifying knowledge flows through the study and modeling of an organization's processes. KoFI focuses on identifying the knowledge required and generated in the principal process activities. KoFI also identifies the location of storage sources and how information can be retrieved from them. This methodology has two main goals: 1) To obtain information that will help to build a knowledge map; and 2) to obtain requirements that will help in the design of knowledge-based systems.

The KoFI methodology consists of 4 phases. Phase 1 consists of identifying the different sources in which knowledge is generated or stored; Phase 2 consists of identifying the types of knowledge used and generated in the organization's principal processes, while Phase 3 identifies how knowledge flows within the organization; finally, Phase 4 consists of identifying the main problems that hinder the flow of this knowledge. These phases will be described in greater detail in the following section.

3. Knowledge flows for interaction in DSD

A study was conducted with a group of distributed software developers from Novutek SC. Novutek is a software factory that bases its production process on the Rational Unified Process (RUP) and uses the Unified Modeling Language (UML) for software modeling and design. Novutek has achieved a SEI

CMMI maturity level 3. The project management methodology used is based on the standards of the Project Management Institute (PMI) and has a group of project leaders in the certification process to become Project Management Professionals (PMP).

The study involved 16 workers, including 4 project leaders, 2 testers, 4 programmers and 6 software architects. The average age of the group was 24.5 years. There were 10 women and 6 men. The average work experience was 2.9 years.

Information was obtained from the software factory by conducting semi-structured interviews and system analysis, as suggested by the authors of the methodology [11, 12].

The interviews were used to understand how developers interact with knowledge sources during working hours. The interviews were conducted individually and were audio recorded. The duration of the interviews averaged 40 minutes. Interviews were conducted in an anonymous format, thus allowing the informants to be as open as possible. Data were extracted from the interviews using open, axial and selective data coding processes, which are used in grounded theory [13].This process began with open coding in which data were divided into categories (recurrent concepts) and subcategories. We subsequently continued with axial coding during which we examined the relationships between categories and the possible causes of their behavior. Finally, for selective coding, the categories were integrated and conclusions were obtained.

Our analysis also consisted of analyzing the technical documentation of systems that were mentioned during the interviews, along with consulting the technical staff of the software factory.

The principal results of these processes are presented in subsections 3.1 to 3.4.

3.1. Phase 1: identifying knowledge sources.

In this phase it was necessary to consider all the knowledge sources that could be used to identify an appropriate moment for initiating a synchronous interaction that would minimize the unwanted consequences of an interruption.

The types of sources that were detected are presented in the following categories:

- *Documents*: this consists of sources that have their origin in physical or digital documents. These include: i) Process: this includes documentation related to the description of the processes of software development. ii) Technical: this refers to specialized documents such as source code, and manuals, among others. iii) Organizational: this includes guidelines towards the conduct that should be followed within the organization.
- *Systems*: this refers to information obtained from or added to the system for software development and project management, and includes: i) Queries: which refers to the search or request for information. ii) Transactional: this refers to adding, deleting or updating information in the system.
- *People*: this refers to all the project collaborators (e.g. programmer, analyst, etc.).

3.2. Phase 2: identifying knowledge topics

In this phase it was necessary to identify the knowledge topics involved in the process of initiating a synchronous interaction in DSD, considering the different types of knowledge generated by the organization. The types of knowledge relevant to our work are related to the characteristics of the software developers' activities.

The activity in this phase does not seek to describe the topics in detail, but to identify them as part of the knowledge requirements. The topics that were indentified in this phase are described as follows:

- *Personal activity management*: this refers to engaging in explicit efforts to maintain control over the participants' commitments, and using strategies to organize, prioritize and focus on what they have and wish to do. It includes the knowledge that people have in relation to the progress of their unfinished activities embodied in documents.
- *Collaborative activity:* this refers to existing knowledge in the project management system concerning shared activities.
- *Coordination of activities:* this refers to knowledge concerning the dependencies required to complete the activity.
- *Communication among collaborators:* this refers to knowledge concerning the means available and the actual communication which takes place to contact a collaborator.
- *Information concerning a suitable contact opportunity*: this refers to the knowledge that people have in relation to a co-worker's current activity in order to initiate collaboration.
- *Software development and project management*: this refers to the knowledge that exists and is generated during the development process. It includes the knowledge that the organization has in relation to the management of the project.

3.3. Phase 3: identifying knowledge flows

This phase involved creating a knowledge flow model of the process for initiating a synchronous interaction in DSD, based on process modeling techniques with a focus on knowledge generation.

That is, where is knowledge generated? Where is it stored? What information sources are involved? Who uses them? It is also important to detect inconsistencies in or problems with the flow of this knowledge.

Figure 2 illustrates some of the knowledge flows that were identified for the analyzed software development activities. It illustrates two subjects working on a related activity of a given project. This is represented by a set of elements that trigger the interaction within an activity (events, people, activities, objectives, actions, resources, etc.). In this case one subject is the issuer while the other is the receiver. It also depicts three main tasks: (1) Identifying the required information about the activities of those involved, (2) Identifying a suitable moment to interrupt other collaborators, and (3) Initiating collaboration if the moment is right. These steps represent the potential for collaboration [14]. That is, during the issuer's activities, s/he needs to contact a colleague (receiver) to ask about certain doubts concerning a common activity. The information regarding the current activity could be shared knowledge for the working group (organizational repository). With this information, the issuer creates knowledge through a cognitive process that includes:

- *Information acquisition*, which is obtained by consulting the activity being performed by the potential receiver (e.g. knowledge about the project activity and role s/he is performing, etc).
- *Knowledge integration*, when the information is obtained from the potential receivers, it is integrated with the tacit and explicit knowledge acquired from other sources (e.g. calendar of pending activities, design documents, project information, own activities, etc.).
- *Decision making*, the integration of knowledge from other sources provides the issuer with greater details about the project activity on which the potential receiver is working on. This allows him/her to determine whether the selected moment is appropriate to initiate synchronous collaboration.

This cognitive process is favored because the knowledge flow could be achieved in a tacit manner by means of a socialization process [11, 15-17], or in an explicit manner by storing knowledge in repositories [15, 18, 19].

3.4 Phase 4: identifying problems in the knowledge flows

This phase consists of identifying and gathering the problems encountered. This allowed the classification process to begin and possible solutions

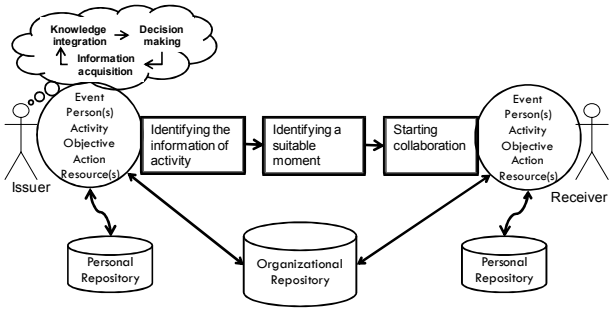

Figure 2. Model of knowledge flows for starting collaboration in rich graphic

to be sought. Problems that may limit or hinder the selected knowledge flows were identified and described.

In the case of DSD, participants are located at remote sites, signifying that the contextual information that is readily available in the co-located case is not accessible, making the communication, coordination and production processes difficult.

The identified problems are described in Table 1. The first column defines the problems that arose from the analysis of the knowledge sources and topics. The second column briefly describes an example situation to illustrate the problem.

As stated by the methodology, the identification of problems allowed us to generate design implications for a communication tool that is better suited to initiating synchronous collaboration in a DSD setting.

Table 1. List of identified problems and situations

Problems	Situation
1. Lack of knowledge about the progress status per activity	During activities, a programmer needs to know if the analyst has begun working on a particular project module. This might help her to organize her pending activities.
2. Lack of awareness of the status of members per activity.	A programmer is unaware of who is working on the same project at a particular moment, and which of them is performing the same activity as her.
3. Coordination problems in common or dependent work units among members of the activity.	A tester does not know why the artifacts are sometimes not sent for review. The tester is therefore contacted by telephone by the person responsible to discover the reason and the date that they will be sent.
4. Initiating an interaction with the right person at the right moment.	Developers use synchronous means of communication to interact (e.g. telephone and instant messaging). However, these means of communication do not provide information about the receiver's activity. It would be useful to have information that would aid in deciding whether the time is appropriate to initiate the interaction.

4. Design implications

Based on the set of knowledge flow problems previously identified (see Table 1), in Table 2 we translated them into features that are later used to identify some design implications for a tool that aims to provide adequate support to initiate synchronous collaboration between DSD workers. These design implications are briefly described below:

1. *Use of technological mechanisms that allow the status of project members and the tasks they are*

Table 2 Design implications

FEATURES	DESIGN IMPLICATIONS
1. Knowledge about the progress status per activity	*I1*. Use of mechanisms that allow the status of project members and the tasks they are performing to be discovered
2. Awareness of the status of members per activity.	*I2*. Use of mechanisms to share and filter project information among colleagues working on related activity.
3. Coordination in common or dependent work units among members of the activity.	*I3*. Use of mechanisms that allow the progress level of the tasks that each member of the project is executing to be discovered. *I4*. Use of mechanisms that allow location of and interaction with members of common or dependent work units.
4. Initiate an interaction with the right person at the right moment	*I5*. Use of mechanisms to identify when one user may interact with another, based on the needs profile, status and activity under execution. *I6*. Services for asynchronous & synchronous communication.

performing to be discovered (I1): technology must provide a mechanism for sharing project information among colleagues or with any other related people, as this will allow the progress status of activities assigned to people and of the project itself to be determined.

2. *Use of technological mechanisms to share and filter project information among colleagues involved in a related activity* (I2): a mechanism is required to ascertain the degree of progress of the work being carried out by each member of the project; this will signify increased information about what is being done in the context.
3. *Use of technological mechanisms that allow the progress level of the tasks that each member of the project is executing to be discovered (I3)*: this requires a lightweight mechanism to consult the degree of progress reported in a particular project activity.
4. *Use of technological mechanisms that allow location of and interaction with members of common or dependent work units (I4*): this requires a mechanism that will allow the location of and interaction with members who share a common or related task, so that they are aware of and anticipate situations that affect the DSD activities.
5. *Use of technological mechanisms to identify when one user may interact with another, based on the needs profile, status and activity under execution* (I5): this requires a mechanism to determine the status of the members of the project and the work they are performing to have a better understanding of the appropriateness of attempting an interaction (i.e. interrupt the worker). A mechanism that identifies when a user may interact with another, based on the interaction need, status and the work being performed both by the worker who wishes to make contact (issuer) and the one being contacted (receiver) is also required.
6. Finally, *technological services for synchronous and asynchronous communication* (I6): synchronous and asynchronous communication mechanisms are required to provide support for both real-time interaction, and for situations in which messages would not be delivered immediately, but stored for later viewing.

5. From knowledge to design concepts

Once the analysis of knowledge flows was completed and the design implications were identified, we proceeded to propose design concepts that would assist us in conceptualizing a solution for the problems identified.

A first concept is that of "Selective Availability", which establishes that a user could be available to interact with other collaborators whose activity is related to the work unit s/he is currently working on (e.g. the status will be online and available), while not being available or even present for others (e.g. online but busy, or offline).

To make this concept operational we examined the concepts of working spheres [20] and potential collaboration [14]. The former allows us to understand how individual workers fragment their work during the workday, and how they manage resources to perform their work, along with how they manage interruptions to their work. However, this concept considers a focus on the workers' individual activities, and lacks a focus on the collaborative part of the work. The latter concept, promotes the way in which information from the individual facet of work, along with information on certain aspects of collaborative work could be shared and used to determine whether a specific moment is appropriate to attempt synchronous interaction to start collaboration, along with specialized spaces for potential and actual collaboration. A proposal that merges both concepts to compose the model of collaborative working spheres is reported in [21]. In our proposal, we introduce the Collaborative Working Spheres (CWS) concept [22] that combines the features of working spheres with that of potential collaboration awareness [14].

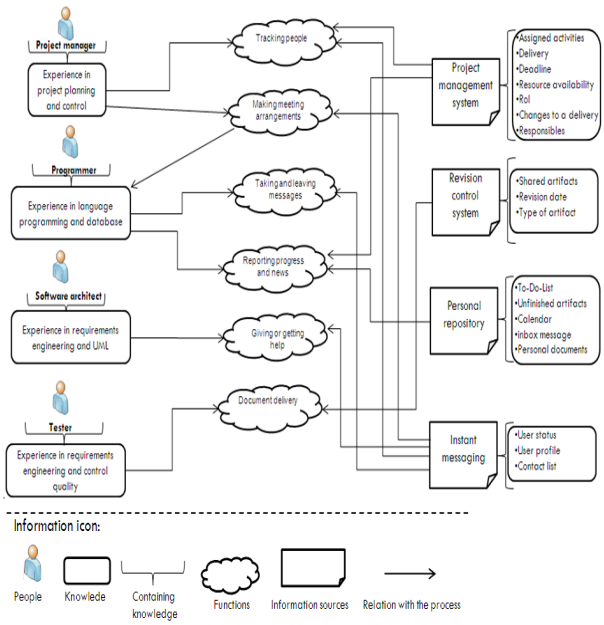

Figure 3. Relationship between knowledge and sources

To make CWS operational, it is necessary to interact with various knowledge sources that are available in the organization. An example of this is shown in Figure 3, in which the developers count on specific knowledge and experience related to their function within the organization. The organization's systems and the work tools (e.g. personal computer, development environment, etc.) similarly contain knowledge which is required at determined moments by work groups. This also illustrates how, during the working day, developers need informal interactions with their colleagues in order to obtain or provide knowledge about their need to locate people, take or leave messages, report advances made in an activity, obtain help or submit a document [23].

Figure 4 presents a summarized map of the knowledge involved to make operational the CWS concept: Based on the interaction purpose, and knowledge required for the initial interaction. This knowledge might be obtained from the worker's activity or from the project specifications, which are the main knowledge sources of the process, and which are in turn classified as documents, people and systems.

5. DSD awareness elements

Taking into account the knowledge map and the design implications previously described, here we present and describe an ensemble of specific awareness elements from a DSD environment [14] which will represent the required knowledge to be

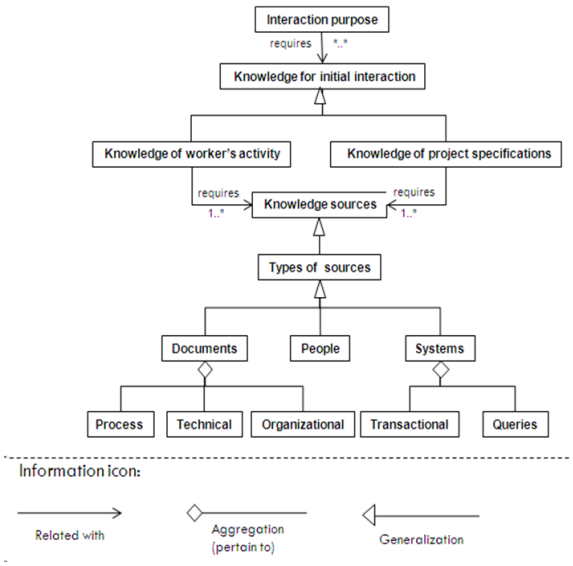

Figure 4. Knowledge map of CWS

gathered and distributed among team members in order to provide support for initiating synchronous collaboration among them in a DSD setting.

A brief description of each information element is presented below:

Resource view: this refers to information concerning the resources of particular interest to the activity or project. This information originates from the project management system repository (system source). This information flow occurs as development stages occur, as indicated by the software development and project management topic. This information is found in the organizational repository (e.g. resource availability, responsible, delivery, etc.).

Gaze: information is gathered from events of the collaborators' common activities. This information originates from the project management system repository (system source). This information flow occurs when system information is added, removed or modified with regard to the project activities. This information is stored in the organizational repository (e.g. completion of an activity of interest, changes to a delivery date, etc.).

Operation: information gathered from the handling of personal documents in a project activity. The flow of this information occurs when developers perform their assigned activities (documents source). This flow originates from personal activity management and activity coordination topics. This information is stored in the personal repository (e.g. relationship of a document with a project activity, document type, etc.).

Notification: information generated from the status change of a project activity. The flow of information originates from interactions between people

performing shared activities and interactions with the project management system. These topics are dealt with in communication among collaborators, software project management and the collaborative activity. This information is stored both in the individual and the organizational repositories (e.g. progress of an activity or project, new activity assigned, etc.).

History: this refers to logged events concerning the activities of particular projects of interest. The history of events originates from the system, people and document knowledge sources. This information is dealt with in the personal activities management and collaborative activities topics and is stored both in the individual and the organizational repositories (e.g. notification from a shared file, inclusion in a given project, etc.).

Presence: this refers to the status of users, which is based on the activity they and their possible contacts are currently working on. This information is obtained from the developers' interactions with their assigned activities (people and document sources). This information is dealt with in topics regarding collaborative activities and the coordination of activities. This information is stored in the personal repository (e.g. available to those who are working on the same activity and unavailable to the rest of the contacts).

Identity: this consists of the personal contact information. This information is obtained from the project management system (system source). This information is dealt with in the software development and project management topics and is stored in the organizational repository (e.g. name, location, role, etc.).

Authorship: this refers to the name of the person assigned to the activities, events, actions, or use of resources, among others. This information is obtained from the project management system and is stored in the organizational repository.

Role: this refers to the profile information or to the type of functions in relation to the task performed by a contact at a given moment (system and documents sources). This information can be obtained from the development and project management system and from documentation from the Human Resources department. It is stored in the organizational repository (e.g. programmer, analyst, tester, etc.).

Teamwork: this refers to groups of contacts automatically created based on the users' current activities (person, system and document sources). This information emerges from people's use of documents and from the operations of the system on the documents. This information concerns the software development and project management topics and is stored in the organizational and personal repositories (e.g. when the user is handling a document concerning project "ERP2Pharmacy", the contact group related to the project is automatically obtained).

To-Do List: this consists of lists of pending activities automatically created with information from the project repository (system source). The information originates from the project management system. This information is dealt with in the personal activity management and software development and project management topics. The information is stored in the organizational repository.

Privacy: Current activity information is exchanged only between contacts that have a common activity (system source). The information originates from the project management system. This information is dealt with in the personal activity management and software development and project management topics and is stored in the organizational repository.

Articulation: this refers to information resulting from an interaction during a project activity (people and document source). This information originates from interactions between developers in shared activities. It is dealt with in the collaborative activities, coordination of activities and software development and project management topics and is stored in both the individual and the organizational repositories (e.g. discussion of a design problem regarding the module "inventories" of the "item queries" activity).

These awareness elements can obtained by be either implicit or explicit mechanisms. *Implicit mechanisms* rely on information collected by the system while collaborators carry out their activities (e.g. the system creates a user profile based on information obtained while the collaborator is working). *Explicit mechanisms* rely on information supplied by actions carried out by collaborators. That is, they require explicit actions from the collaborator (e.g. the user explicitly provides the information that is required to create a user profile).

Once this ensemble of awareness elements had been established, it was necessary to determine the type of technological mechanisms (GUI elements) to be used to provide the information element in a prototype tool. In the following section we describe each of the components proposed for the user interface of the proposed prototype.

6. Prototype

Based on the identified design implications, and on the resulting design concepts, and using the proposed awareness elements, we designed and developed a prototype tool to provide support for initiating collaboration in a DSD setting. This tool

was named CWS-IM (Collaborative Working Spheres – Instant Messenger).

This section describes how the previously detected awareness elements were implemented in the developed prototype.

Table 3 Components associated to information elements

GUI Component	*Information element*
• Contact list	• Presence, Role, Teamwork and Gaze
• Identity Panel	• Identity and Gaze
• Contact view	• Presence and To-Do List
• Contact information ToolTip	• Authorship, Identity, Privacy and Operation
• Location label	• Identity and Location
• Notification history	• History, Notification and Gaze
• Task Panel	• To-Do List and Resource view

Table 3 presents the information elements identified along with the associated GUI component of the prototype. Figures 5 and 6 show the actual look of the interfaces proposed for the CWS-IM tool. A brief description of them is presented below:

- *Contact list:* the GUI of CWS-IM presents a list of potential collaborators and of their assigned activities in a particular project (see 5A and 5F). The contact list can be used to initiate a chat session with a group of collaborators or with one collaborator. This component allows users to obtain information concerning the other workers' presence, role and teamwork at a glance.

- *Identity panel:* this is used to show the actual collaborator's information such as name and role (5B). This component also allows users to obtain information on their own identity at a glance.

- *Contact view:* this option allows users to choose which view they prefer (5C), and also permits them to obtain a list of contacts per project (5A) or a list of the assigned activities of the user in the project (5F).

- *Contact information ToolTip:* this is accessed through a right click over the desired contact (5D). This component shows current information about potential collaborators, such as authorship identity and current activity. The information of the potential contact is accessible only if they have common activities or projects (privacy).

- *Location label:* This component provides information about the person's current geographical location (5E). This information is supplied based on the assigned IP address. It allows information complementary to identity to be provided.

- *Notification history* (6A): This component allows users to obtain the history of events regarding the projects of interest and notifications regarding assigned activities at a glance.

- *Task panel* (6B): This component has been added to show the different perspectives of a task that is being carried out (e.g. which project or activity is under execution, which resources have been linked, etc.).

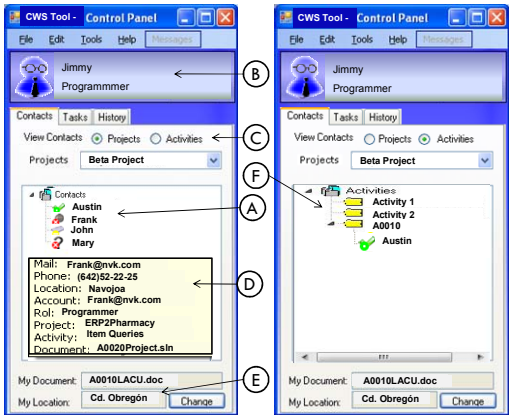

Figure 5. Contact list interface

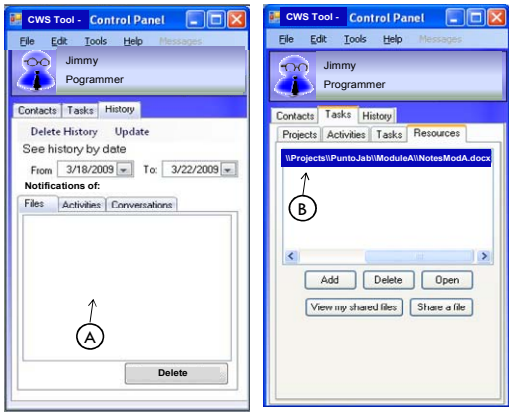

Figure 6. Notification history and Task panel interfaces

Furthermore, in order to implement the *Selective Availability* concept, the tool implements a set of rules for notifications, so that if a user needs to contact a collaborator, the tool attempts to guarantee that the person that is being contacted is working on a common activity or project.

To achieve this, the status of a user is represented in the GUI by using a color code (see Figure 7): if the status of the user is presented in Green, this means that the users (issuer and possible receiver) are working on the same project and on the same activity (Available); if the status is presented in Yellow, this means that the users are working on the same project, but on different activities (Reachable, but busy); and if the status is presented in Red, this means that they are working on different projects (Busy). With this, we aim to provide information elements in a distributed situation to enable an awareness level similar to that which is available to an issuer in a

situation of co-location; for example, by simply observing the potential recipient.

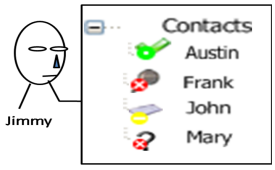

Figure 7. Color Code

Thus, CWS-IM seeks on the one hand, to permit the handling of potential collaboration for synchronous situations such as: i) having knowledge on the availability of developers who are connected and the role they are playing at that moment; ii) having knowledge on the assigned work units to work individually and in groups; iii) notifying involved teammates when a unit of work finishes, and iv) allowing the reception of implicit notifications concerning the work units in which the worker is involved on. On the other hand, for past or future situations, to allow the handling of potential collaboration for asynchronous situations, such as v) having knowledge on the state of a previously worked work unit, and vi) obtaining information on work units likely to be performed.

7. Related work

Similar projects that focus on providing information regarding the activities that occur in a development group have been proposed. For example: Palantir [24] is a system that complements existing configuration systems by providing distributed awareness of the project's progress. This is done through a graphic display which shows measurements of severity and impact of the changes in the artifacts, so that developers could anticipate the problems that may result from those changes. This project however is focused only on the management of changes to the artifacts. ProjectWatcher [25] is a system designed to provide awareness support regarding "who is who in general" and "who works in this area of the code". There are also mechanisms (e.g. Eclipse Plugins [26]) to inform users about the location of a concurrent change at the class, method and line level. These two projects allow keeping the team informed about the state of specific code sections and files In contrast, our proposal provides a new way to automatically reflect the change in status of a user according to assigned project activities, as well as status filtering according to the activity of both the issuer and the receiver (Selective Availability). Further, none of the cited works focuses on informing the group with the dynamicity and granularity level regarding the activity that individuals perform from their own workspace, the artifacts used in this activity, and the related information of the associated project, and involved team members.

Finally, it is important to highlight that the approach and ensemble of awareness information elements and mechanisms to gather and present them does not intend to be conclusive nor comprehensive, but a starting proposal from which to explore additional elements and mechanisms that facilitate starting an interaction that is better informed and suited to the needs of both the issuer and the receiver.

8. Conclusion and future work

Geographical distribution in DSD implies that people do not have the opportunity for face to face interaction, which limits the ways in which people gathers and use information to synchronously contact others at timely opportune moments. In this work, our premise was to take advantage of existing tacit and explicit knowledge of workers in a DSD organization to facilitate their starting collaboration in a more appropriate and informed manner. Our approach to cope with this allowed us to identify documents, people and system as sources of valuable knowledge; from where information elements such as own and collaborators presence and identity, artifacts in use, people and resource location, associated projects, and plans and history logs, among others, could be technologically retrieved and provided to overcome physical distance, the main barrier that prevents this knowledge to flow among issuers and receivers of potential interactions in DSD. Once this information was identified, we proposed a set of design implications for a tool that aims to provide adequate support to initiate collaboration between DSD workers.

Based on these insights, an ensemble of information elements and gathering and presentation mechanisms were proposed to develop CWS-IM, an extended IM application. CWS-IM could improve the way in which DSD workers start collaboration compared with traditional instant messaging because it:

- distributes explicit organizational knowledge (e.g. project status information) in real-time
- provides explicit information through specialized mechanisms to specific project- and activity-related team members;
- offers an individual perspective to the DSD group (e.g. availability of information regarding each co-worker related to the project according to the assigned work and the activity currently under execution);
- provides a group perspective regarding a specific team member (e.g. it notifies whenever

a developer is available to certain project-related members, and unavailable to others).

Concerning future work, the use of CWS-IM is going to be evaluated in a real situation by deploying it for a test at the software factory where the preliminary study was performed.

Acknowledgments

This work has been funded by the PEGASO/MAGO project (Ministerio de Ciencia e Innovación MICINN and Fondos FEDER, TIN2009-13718-C02-01). It is also supported by MEVALHE (HITO-09-126) and ENGLOBAS (PII2I09-0147-8235), funded by Consejería de Educación y Ciencia (Junta de Comunidades de Castilla-La Mancha), and co-funded by Fondos FEDER, as well as MELISA (PAC08-0142-3315), Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia in Spain. This work was also supported by UABC under grant 207 of the Convocatoria Interna de Proyectos de Investigación; and by Asociación Mexicana de Cultura A.C.

References

[1] R. Prikladnicki, J.L.N. Audy and J.R. Evaristo, "Distributed Software Development: Toward an Understanding of the Relationship between Project Team, Users and Customers," *Proc. 5th Int'l Conf. Enterprise Information Systems (ICEIS 03)*, 2003, pp. 417–423.

[2] M. Cataldo, M. Bass, J. D. Herbsleb and L. Bass, "On Coordination Mechanisms in Global Software Development," *Proc. The international Conference on Global Software Engineering* IEEE Computer Society, 2007, pp. 71-80.

[3] J.D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *Software Engineering*, vol. 29, no. 6, 2003, pp. 481-494.

[4] D. Damian and D. Moitra, "Guest editors' Introduction: Global Software Development: How far have we come?," *IEEE Software*, vol. 23, no. 5, 2006, pp. 17-19.

[5] U. Bellur, "An academic perspective on globalization in the software industry," *Proc. The 30th annual international computer software and applications conference (COMPSAC'06)*, 2006, pp. 52-54.

[6] M. Czerwinski, E. Horvitz and S. Wilhite, "A diary study of task switching and interruptions," *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 175-182.

[7] S. Fussell, S. Kiesler, L. D. Setlock and P. Scupelli, "Effects of instant messaging on the management of multiple project trajectories," *Proc. The SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 191-198.

[8] J. Ellis and L. Kvavilashvili, "Prospective memory in 2000: Past, present and future directions," *Applied Cognitive Psychology*, vol. 14, 2000, pp. 1-9.

[9] L. Dabbish and R.E. Kraut, "Controlling interruptions: awareness displays and social motivation for coordination," *Proc. The 2004 ACM Conference on Computer Supported Cooperative Work* 2004, pp. 182-191.

[10] R. E. Kraut and L.A. Streeter, "Coordination in software development," *Commun. ACM* vol. 38, no. 3, 1995, pp. 69-81.

[11] O. M. Rodríguez-Elias, A. Vizcaíno, A. I. Martínez-García, J. Favela and M. Piattini, "Knowledge Flow Identification," *Encyclopedia of Information Science and Technology*, 2nd ed., Information Science Reference: IGI Global, 2008, pp. 2337-2342.

[12] O. M. Rodríguez, A. I. Martínez-García, A. Vizcaíno, J. Favela and M. Piattini, "A Framework to Analyze Information Systems as Knowledge Flow Facilitators," *Information and Software Technology*, vol. 50, 2008, pp. 481-498.

[13] Anselm Strauss and J. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques / A. Strauss, J. Corbin*, Newbury Park, EUA : Sage, 1990.

[14] A.L. Morán, J. Favela, A. Martinez and D. Decouchant, "On the Design of Potential Collaboration Spaces," *The International Journal of Computer Applications and Technology (IJCAT)* vol. 19, no. 3/4, 2004, pp. 261-271.

[15] I. Nonaka and H. Takeuchi, *The knowledge-creation company: How japanese companies create the dynamics of innovation*, Oxford University Press, 1995, p. 304.

[16] T. H. Davenport and L. Prusak., *Working knowledge: How organizations manage what they know*, Harvard Business School Press, 2000.

[17] I. Oshri, J. Kotlarsky and L.P. Willcocks, "Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects," *J. Strateg. Inf. Syst.*, vol. 16, no. 1, 2007, pp. 25-49.

[18] O. M. Rodríguez, A. I. Martínez, J. Favela, A. Vizcaíno and M. Piattini, "Understanding and Supporting Knowledge Flows in a Community of Software Developers,"3198, L. N. i. C. Science, ed., Springer, 2004, pp. 52-66.

[19] K. Dalkir, "Knowledge Management in Theory and Practice," *Elsevier/Butterworth Heinemann*, vol. 13, no. 2, 2005; DOI http://doi.acm.org/10.1145/1116715.1116753.

[20] V. Gonzalez and G. Mark, "Constant, Constant, Multi-tasking Craziness: Managing Multiple Working Spheres," *Proc. The SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, 2004, pp. 2004.

[21] R. R. Palacio, A. L. Morán and V.M. González, "CWS: An Awareness Tool to Support Starting Collaboration in Global Software Development," *The Open Software Engineering Journal. Special Issue on Global Software Development and its Challenges*, 2009.

[22] R. R. Palacio, A. L. Moran, V. M. Gonzalez and A. Vizcaino, "Collaborative Working Spheres as support for starting collaboration in distributed software development," *Proc. 13th International Conference on Computer Supported Cooperative Work in Design, 2009*, , pp. 636-641.

[23] E. Isaacs, S. Whittaker, D. Frohlich and B. O'Conaill, *Informal Communication Re-examined: New Functions For Video in Supporting Opportunistic Encounters*, Lawrence Erlbaum, 1997.

[24] A. Sarma and v.A.d. Hoek, "Palantír: Increasing Awareness in Distributed Software Development," *Proc. Workshop on Global Software Development (ICSE 2002)*, 2002, pp. 28-32.

[25] C. Gutwin, K. A. Schneider, D. Paquette and R. Penner, "Supporting Group Awareness in Distributed Software Development," *Proc. EHCI/DS-VIS*, 2004, pp. 383-397.

[26] C. Ignat and G. Oster, "Awareness of Concurrent Changes in Distributed Software Development," *Proc. The OTM 2008 Confederated international Conferences, Coopis, Doa, Gada, Is, and ODBASE 2008. Part I on on the Move To Meaningful internet Systems*, Eds. Lecture Notes In Computer Science, 2008, pp. 456-464.