Evaluating Awareness Information in Distributed Collaborative Editing by Software-Engineers

Julia Schenk
Institute of Computer Science
Freie Universität Berlin
Berlin, Germany
julia.schenk@fu-berlin.de

***Abstract*—In co-located collaborative software development activities like pair programming, side-by-side programming, code reviews or code walkthroughs, the individuals automatically gain a fine granular mutual understanding of where in the shared workspace the other participants are, what they are doing and what their levels of interest are. These points of so called awareness information are critical for an efficient and smooth collaboration but cannot be obtained via the natural mechanisms in virtual teams. Application sharing and groupware for collaborative editing are widely used for collaborative tasks in distributed software development but considered from the awareness and flexibility aspect they are far off the co-located setting. To better support virtual team collaboration by improving tools for distributed software development it is neccesary to evaluate awareness and its impacts to certain collaborative situations. Awareness itself is an invisible phenomenon and due to its intangible nature cannot be easily observed or measured. Thus we recorded virtual teams using Saros, a groupware for distributed collaborative party programming, respectively VNC and now analyse these videos using the grounded theory methodology. This approach for evaluating awareness leads to various problems concerning the recording setup and time exposure for analysis.**

***Keywords*-Workspace Awareness; Distributed Software Development; Collaborative Editing; Groupware; Application Sharing; CSCW;**

I. Awareness during Collaborative Software Development

Agile software development practices like Pair Programming or Side-by-Side Programming [1] emphasize the importance of the keen communication and interaction of the parties concerned. The participants sit next to each other and share a single monitor, one set of input devices, and a common view on an artefact they are collaborating on. In this face-to-face setting the individuals automatically gain a fine granular mutual understanding of what the other one is doing in the general physical environment as well as on the shared artefact.

This mostly unconscious gained information is refered to as **awareness information** and entails "an understanding of the activities of others, which provides a context for [one's] own activity" [2]. Gutwin et. al [3] argue that the success of a collaboration hardly depends on the awareness about others and their interactions with the shared workspace. The authors in [4] divide the awareness information that is relevant during a collaboration into four main categories:

- Informal Awareness: "is the general sense of who's around and what they are up to"
- Social Awareness: "is the information that a person maintains about others in a social or conversational context: things like whether another person is paying attention, their emotional state, or their level of interest."
- Group-Structure Awareness: "knowledge about such things as people's roles and responsibilities, their positions on an issue, their status, and group processes."
- Workspace Awareness: "up-to-the minute knowledge about others' interaction with the task environment."

When conducting code reviews, performing code walkthroughs, introducing newbies or doing collaborative programming tasks, workspace awareness in particular is a critical factor because it gives indication of "where in the space others are working, what they are doing, and what changes they are making" [3]. But when teams are not working co-located workspace awareness information cannot be obtained in the natural way. For that reason the next chapter discusses the implications of workspace awareness in distributed teams.

II. Approaches for Distributed Collaborative Software Development

Virtual teams face the problem that they cannot use their natural interaction and information gathering mechanisms. Groupware supporting real-time collaboration is still in the early stages of development and lacks a lot of awareness information due to a) the technical limitation of picking up awareness information only through input devices [5], b) the lack of knowledge about how to present it in an undisturbing but effective way and c) the complex and intangible nature of awareness information.

As a consequence, virtual teams face the problem that they do not collaborate as effectively and efficiently as their collocated counterpart [6]. The next section describes two

approaches for virtual teams to do software development activities cooperatively.

A. Application Sharing

Companies with distributed locations or outsourced respectively offshored development departments widely use application sharing for their collaborative tasks. Working together via application sharing has several advantages regarding workspace awareness since both participants can see the same screen:

- They are aware of each other's viewport.
- They are always in the same artefact and thus know where each one is working in the shared workspace.
- They directly see what the other one is doing in the shared workspace.
- They can make use of deictic references to support verbal communication.

But there are also some drawbacks concerning the flexibility during the collaboration:

- The participants compete for the input devices and cannot use them independently. That is also a reason why collaboration via application sharing does not scale well for more than two parties.
- The participants have to reside in the same artefact; they cannot alternate between exploring separately, working independently, and finding together again.
- When sharing an application, e.g. a webbrowser with open tabs, there are also privacy issues to consider.

B. Saros - Distributed Party Programming

Since 2006, our research group develops Saros[1] which is an Eclipse[2] plugin for distributed collaborative software development, in particular collaborative editing. The concept of Saros is that in a session arbitrarily many distributed team members work together on one or more fully or partially shared Eclipse projects. All participants of the session have a local copy of the shared projects and Saros keeps these in sync while the participants concurrently edit the same or different files.

As shown in Figure 1 Saros provides some relevant awareness information like who is online, who is working on which file, what each participant is doing in a file and who can see which portions of a file. Regarding the coupling of the team members Saros provides a better flexibilty than application sharing, since in a session the team members can work independently and explore separately in the shared workspace while still being able to find together for problem solving or collaborating on a task. Apart from the awareness information in the Eclipse project explorer that marks open and active files from the other participants, a double-click on a buddy in the buddy list enables a jump to the position of this buddy in the workspace. Features like VoIP, Screensharing and a collaborative whiteboard are under active development. Nevertheless tests with users showed that the existing awareness information in Saros is not perceived as intended and that Saros also lacks some relevant information because Saros shares only the Eclipse editor. Information from the console, the project explorer, java applications that were built and ran from Eclipse or the webbrowser, which is important in case of web application development, are not shared.

[1] http://www.saros-project.org

[2] http://www.http://eclipse.org/

C. Application Sharing versus Saros

Since application sharing shares an application in its entirety, the users benefit from the better workspace awareness than they currently do with Saros. In contrast Saros allows for different collaboration styles whereas application sharing restricts the users to only work in the same file.

We will discuss a research problem arising from these observations concerning awareness and flexibility in the next section.

III. Research Problem

We are not aware of any empirical work in the software engineering domain that considers the aspects of awareness during the stated approaches for virtual team collaboration and discusses the awareness provided by application sharing versus the better flexibility in a groupware like Saros. With Saros as a testbed we do not want to draw a general good - bad comparison between application sharing and Saros, but we want to examine where each approach concerning awareness and flexibility has its advantages in certain situations dependent on factors like kind of task and the mode of collaboration. The research problem we will contemplate on has two apects. First it pertains the question in which situations the two aspects flexibility and awareness of each approach are dominant and thereby relevant for an effective and smooth collaboration. After examining these aspects a second question of interest is how to better balance or combine the particular strengths in a groupware like Saros. Awareness by nature is a mostly unconscious and invisible phenomenon and therefore the next section considers the issue of evaluating awareness during distributed collaborative editing by software engineers.

IV. Evaluation of Awareness in Distributed Software Development

In our understanding awareness by itself cannot be directly observed or measured but only events that are the results of the presence or absence of awareness can be observed. Due to this nontrivial problem concerning the evaluation of awareness the next section lists different methods and discusses their suitability for evaluating awareness.

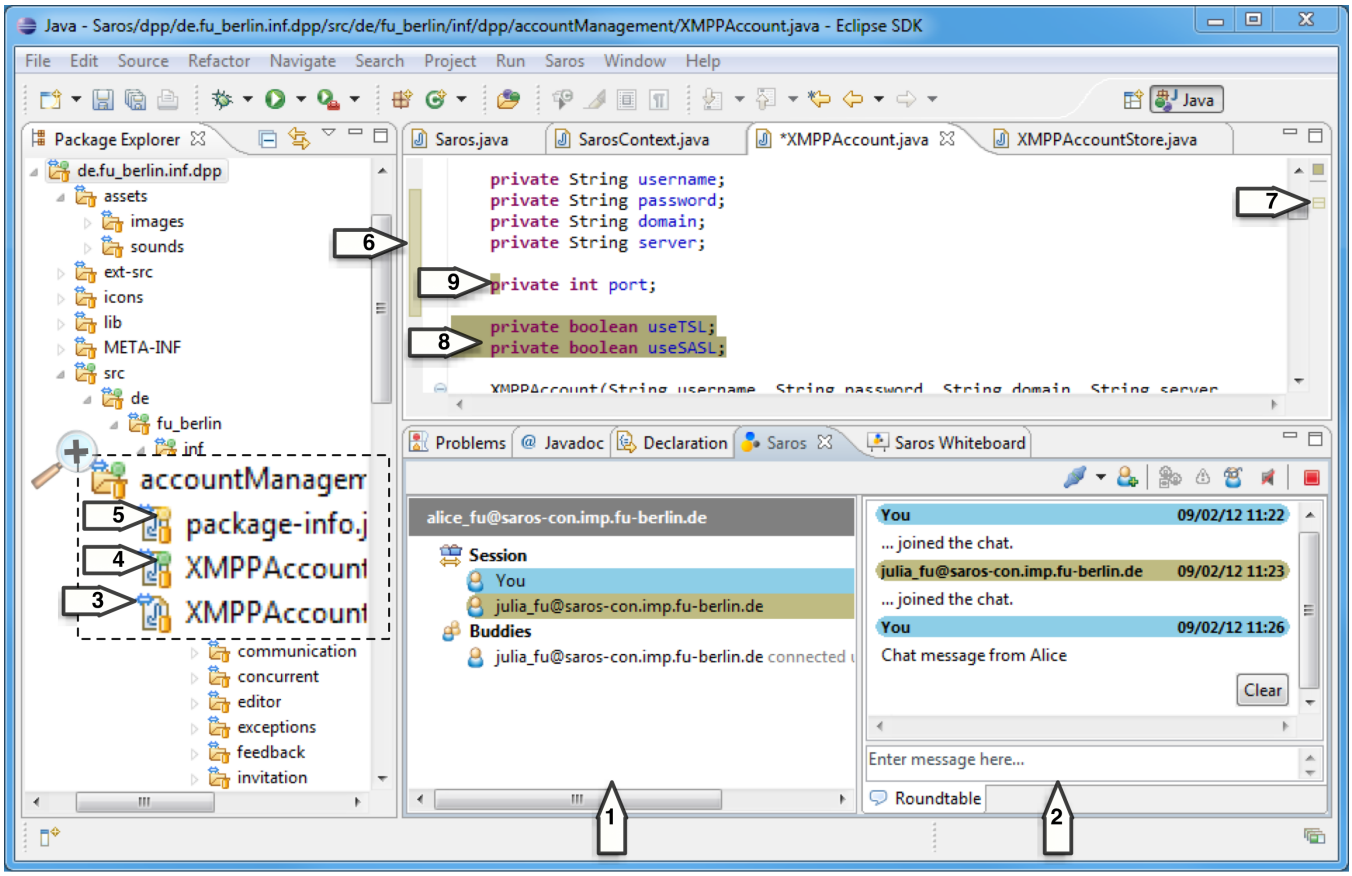

Figure 1. 1) The buddy list: Each Participant has a list of contacts. From online contacts one can choose with whom to collaborate. During a session each participant is assigned a unique color to match the displayed awareness information. 2) During a session the participants can communicate via a roundtable chat. 3) Shared files and folders are annotated with a double arrow symbol. 4) A file annotated with a green dot indicates that this file is in foreground by any other participant of the session. 5) A file annotated with a yellow dot decorates all files that are open in the editor view of any other participant but currently not in foreground. 6) A colored bar at the left side of the editor shows which lines of the current viewport the other participant assigned with this color can see. 7) A bar next to the scrollbar on the right of the editor marks the position of the other participants in the file. 8) If a user writes or selects text, the selection respectively the last 20 characters written by this user are highlighted in the user's colour and visible for all other participants. 9) The cursor position of each user in the shared artefact is highlighted in the user's colour.

A. Discussion of Methods for Evaluating Awareness

Several methods to evaluate awareness are conceivable: In **Interviews** and **questionnaires** the participants would have to bring in mind information they automatically and unconsciously perceive and use. Only few users are able to reflect such information and we do not have a user group that is large enough to get a representative sample out of it. Another issue concerning the design of the questions arises from the nature of the problem: We do not have enough understanding of awareness and its impact in a distributed collaborative situation and hence do not know what information to ask for. Semi-structured or open interviews as well as open questions in questionnaires could be helpful to get insight into the participant's emotions, experience and uncertainties but they are not adequate as a sole data source for evaluating awareness.

Due to the distributed setting of the research problem when **observing participants**, the researcher can only be present at one side of the collaboration. Because of the onesided perception it is very likely that valuable information is missed: it does not provide a holisitc view of the situation or insight as to why certain things happen or why participants behave the way they do and there is no data available to do a post-analysis of the context.

Another possiblity to monitor what the participants do when they collaborate via application sharing or Saros is to **equip their workspace with instruments**. This enables data-collection from all participants but the generated data is limited to record the participant's interactions with the workspace via their input devices and has not enough substance to be semantically sufficient.

Video recording provides the possiblity to capture a collaboration situation in its entirety. Therefore it is neccessary

to capture the workspace as well as at the nonverbal (gesture and mimic) and verbal communication of all concerned parties. The physical distribution of the participants leads to nontrivial issues concerning the recording infrastructure as well as the consolidation and synchronisation of the video and audio data. A post-analysis of video recordings is very costly in terms of time. However, we believe that the information about what is happening in each participant's workspace, their verbal communication, and nonverbal hints from gestures and facial expressions is valuable data in order to get an understanding of awareness and its impacts on collaborative editing. It is important to get an understanding of where and why from the awareness point of view their collaboration works well, where difficulties arise and how the participants handle or bypass them, particularly where verbal or written communication substitues natural awareness mechanisms.

Our goal for the workshop is to exchange opinions and discover new approaches concerning the evaluation of awareness in distributed collabroative editing. We also would like to discuss our method of qualitative post-analysis of video recordings. Therefore the next section will give an explanation of the considerations and experiences we have so far concerning the post-analysis of captured collaborations.

B. The Technical Approach for Evaluating Awareness

In our setting the participants among themselves collaborate via VNC[3] respectively Saros, communicate via a voice over IP connection and have a webcam installed. For a holistic analysis of the collaboration we synchronize the recordings of the participants and merge them into one video. The physical distribution of the participants implicates several problematic aspects for the data recording where we elaborated two approaches: Either screen, video and audio are recorded on each participant's local machine using a screen recording software or each participant connects to a host machine where the recording of both participants happens central at the same time.

For the VNC setup we used the central recording approach and it worked well. The local recoding on the participants' machines lead to problems since on the participants' machines the audio and video signal ran out of sync. We had to sync the signals for each participant and then consolidate and sync the single videos into one.

C. The Methodical Approach for Evaluating Awareness

Awareness during a collaboration only manifests through symptoms of the presence or absence of awareness. Our assumption is that for evaluating awareness during distributed collaboration situations it is neccessary to **gradually get an understanding of the phenomena indicating the presence or absence of awareness and to consider them in their context** in order to evaluate their implication for a certain situation. The understanding of the symptoms of awareness and their impact to the effectiveness of distributed software develpoment is a sociological phenomenon in a technical context. This area of research is relatively unexplored and little is known about how to grasp awareness information. We want to be open-minded and want do develop an understanding of awareness that emerges from the considered data. That is why we think the grounded theory methodology is well suited to investigate and evaluate awareness information. During the research process it will become clear what is relevant in the region under examination. Data sources are video recordings of virtual teams collaborating via application sharing or a Saros. To sanity-check the observed phenomena and to clarify them as well as to gain deeper insights into what happens in the participants' heads, open or semi-structered interviews will be conducted shortly after the recording. The research process is guided by systematic elicitation and analysis of the video recordings but we are also aware of the scale-problem concerning the analysis and therefore the risk of not being able to analyse enough material. That is why one of our goals for the workshop is to discuss and get ideas how to improve or optimize the evaluation of awareness in distributed collaborative editing by software engineers.

[3] http://www.realvnc.com/

REFERENCES

[1] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Longman, 2004. [Online]. Available: http://www.amazon.de/dp/0201699478/

[2] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. New York, NY, USA: ACM, 1992, pp. 107–114.

[3] C. Gutwin, S. Greenberg, and M. Roseman, "Supporting awareness of others in groupware," in *CHI '96: Conference companion on Human factors in computing systems*. New York, NY, USA: ACM, 1996, p. 205.

[4] ——, "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation," in *HCI '96: Proceedings of HCI on People and Computers XI*. London, UK: Springer-Verlag, 1996, pp. 281–298.

[5] C. Gutwin and S. Greenberg, *The Importance of Awareness for Team Cognition in Distributed Collaboration*, 2004, pp. 177–201.

[6] G. M. Olson and J. S. Olson, "Distance matters," *Hum.-Comput. Interact.*, vol. 15, no. 2, pp. 139–178, Sep. 2000. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI1523_4