SCI: Towards a Social Collaborative Integrated Development Environment

Hani Bani-Salameh
University of Idaho
hsalameh@vandals.uidaho.edu

Clinton Jeffery
University of Idaho
jeffery@uidaho.edu

Jafar Al-Gharaibeh
University of Idaho
jafara@vandals.uidaho.edu

Abstract

Software development teams face challenges communicating with each other. This paper presents the design of a social real-time collaborative IDE called SCI that unifies the concepts of social network and collaborative IDE. SCI integrates presence and activity awareness information and collaborative program development tools. Activity awareness information provides a sense of the presence of each team member: their activity on joint projects, technical interests, currently active sessions, and availability profile. The collaboration tools provide a wide range of facilities for synchronous and asynchronous collaboration and information sharing between team members.

1. Introduction

Software developers routinely work in teams. While some projects involve only immediate team members, many projects involve a broader community of individuals from different institutions [1]. Many developers contribute to several projects in any given week. One of the pervasive challenges facing any software development team is getting the right level of communication to coordinate their work and perform their tasks effectively, and this problem is more difficult for distributed teams.

Integrated development environments allow developers to solve design and coding problems and create software products. During these activities, developers interact with people who share interests in particular technical subjects. Social networks build online communities of people with common interests. Our hypothesis is that IDEs and social networks are complementary in software development, because software development communities are social networks. SCI (Social Collaborative IDE) is a tool that integrates development environment and social network. This paper presents SCI and describes the interaction between social networking and collaboration tools.

Large scale software development is a social endeavor. For team members to function effectively, they must maintain a certain level of social awareness. Developers must be aware of their other team members' roles and activities, as well as the resources in the community relevant to a given project. Also, in order to catch people when they are "in" and focused on a given task, developers need to be aware of the other team members' primary activity windows and/or activity history [2].

The present work started with a synchronous collaborative integrated program development environment called ICI (Idaho Collaborative IDE) that provides a set of real-time collaboration features for developers. ICI is well-suited for scheduled interactions between people who already know each other and work together. In order to support communication and collaboration between less cohesive development groups whose community members are geographically distributed, substantial additional capabilities are needed.

Consider a niche programming community, such as the one for which SCI was developed. Niche developer communities are common, often a few dozens, hundreds or thousands of developers whose work uses particular software tools such as domain-specific languages, library API's, or open source software projects. Supporting such a community requires more than the ability to e-mail, leave each other messages, or post a comment to the project mailing list. Developers need the ability to find others with common interests or needed expertise, and ask for their assistance easily. The SCI project extends ICI to support asynchronous collaboration and community activity integration, increasing the developers' awareness of each other, and of artifacts, resources, and activities, and encouraging team communication.

2. Overview of ICI

ICI is a collaborative IDE integrated inside the CVE virtual environment (cve.sf.net). CVE is a multi-platform collaborative virtual environment where users interact in a 3D virtual world. The virtual environment provides developers with a general view of other users and what they are doing. It allows developers to chat via text or VoIP with other team members and with developers from other teams in real time. Users may invite one another into collaborative IDE sessions, where they work together on tasks such as program design, coding or debugging.

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.420

IEEE computer society

ICI's architecture consists of four major components: 1) a collaborative editor (C) supports shared source code editing and navigation, 2) a collaborative shell (D) allows shared compiles, program runs, and real-time debugging sessions, and 3) a set of communication tools for text and voice chat (A), provided by the virtual environment, and 4) an interface for collaboration control (G), which allows users to invite other developers (F), take turns at the IDE controls, and enter and leave the collaborative session (E). ICI supports the languages C, C++, Java, and Unicon.

SCI enhances ICI's collaboration features (Figure 1). In the CVE virtual environment, users interact within a 3D virtual world. ICI extends CVE with synchronous tools to communicate, interact, and collaborate on programming. SCI extends ICI with group, user, project, and session presence and awareness information. SCI's asynchronous tools drive the use of ICI's synchronous tools, and the two categories complement each other. The result is a unique development environment.

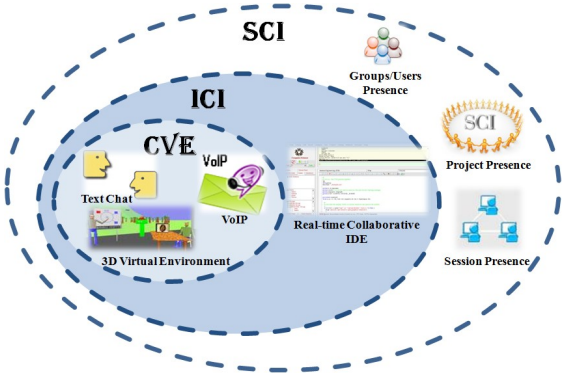

Figure 1. The SCI architecture

3. Awareness in Software Development

Combining a development environment with presence, social activity, and community integration begins with the awareness requirements for distributed development. The identified issues were: the kinds of awareness developers need, the sources of information available, and how awareness about team or community members changes over time. In general, distributed developers maintain general awareness of the entire team, as well as detailed awareness of people that a developer is working with, or asks assistance from. Developers maintain awareness primarily through text-based communication tools, such as: mailing lists, email systems, and chat systems.

Developers maintain awareness of people working on their project, in what parts of the code others are working, and their areas of expertise. They obtain this information by: (1) watching the team/group news feed; (2) observing who changes the group projects artifacts; and (3) by observing changes to the projects artifacts and CVS commits (users get it by email). This helps developers to be up-to-date about both changes to the project artifacts and the activities of other distributed team members.

4. SCI Design

SCI provides access to as much developer community information as possible, and makes collaboration more tangible within the community. As teams become bigger, the social challenges of coordinating have a greater impact on the success of the project. SCI provides community social network features that: 1) facilitate communication between groups of developers who are related to each other by interdependencies such as: a special interest, work, ideas, or perhaps friendship; and 2) assist developers in navigating the SCI environment, and make it easy to find appropriate project partners.

SCI is composed of two parts: 1) an information part that providing presence and activity awareness information about each team member, their activity in various projects, tasks assigned to each member, and progress on these projects; 2) a tools part, where SCI provides synchronous and asynchronous tools to facilitate communication, collaboration, and information sharing.

4.1. SCI Architecture Components

SCI runs on Microsoft Windows, Linux, and Mac OS. Its major components (Figure 2) are: collaboration spaces (A), text and voice chat (B), news feed (C), session tree (D), users tree (E), groups tree (F). D-F provide awareness information that users see while working on their projects.

Tabs (C), appears to the left side of the figure showing social awareness of the users, their status (online, offline, or idle), active collaborative sessions and members of each session. It is showing the available teams, the virtual rooms, and who belongs to each team. Also, it is showing the presence of each team member, their activity in the project, tasks assigned to each member, progress on these parts of the project, and their activity history.

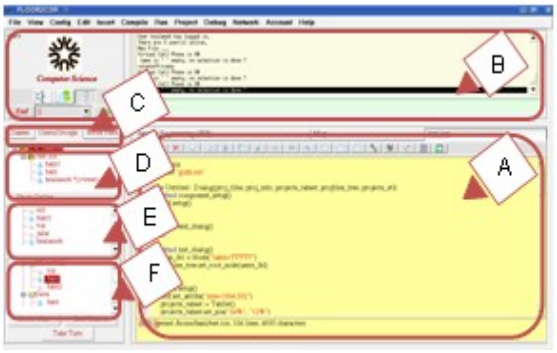

Figure 2. The SCI environment.

Within SCI, the social networking components include elements that would otherwise be served by standalone tools that provide little awareness information:

- ***Email:*** SCI includes public and private e-mail, including group e-mail tied to groups' news feeds.
- ***Special Interest Groups (SIGs):*** developers join SIGs to interact on shared interests. SCI's original SIGs are for Java, C++, Unicon, and Software Engineering. A smart NPC (Non-Player Character) hosts each SIG's virtual room. The NPC offers help and gives materials related to the SIG. Constructing the NPCs' expert-system rules for user interaction is a collaborative activity of the SIG. Users can interact with the NPC when nobody else is available to help.
- ***Profiles:*** Users view others' information within their community circle. The user profile shows friends, groups, projects, and a mini-feed that shows the user's activities and recent events. The profile includes a calendar to inform everyone of project deadlines and user availability. The local time is given so that other developers can be reminded of time zone differences.

Users can view other members' personal information, projects, talks, and any other content they want to display. The question here is: why is there a need for such information inside a community where all members are professionals? The value of displaying users' personal interests along side with the work information is because collaboration is between people, and knowing about the people you are working with, especially those who are distributed all around the globe, can be a key for building effective teams. Figure 3 shows a screen shot of the user profile design.

- ***News Feed and discussion threads:*** The News Feed highlights new projects, changes to projects, new groups, members who have joined groups, active debugging or editing sessions, and other updates. The News Feed also shows conversations taking place between users. Each group has its own discussion page, and all users in the group can chat with all other members in that group, or view and send emails to the discussion feed in the room's page [5]. A Personal News Feed shows updates tailored to that user, allowing developers to track changes in projects and team member activity. Users control what types of information are shared automatically with friends. Feeds provide the user with social awareness such as: the status of the community, number of online users, their availability, and the active projects.

To make it easier for developers, automated summaries of news feeds are supported. A developer can define the priority for new entries (low, medium, or high) and set access permissions. Private denotes information for the friend's circle only; Public can be accessed by everybody. Each project has a mini-feed that can be accessed by the project team members. Groups also have mini-feeds. In all cases, users designate which specific user or list of users can access posted information, if they so choose, so that groups can maintain privacy.

The system provides developers with automatically generated postings about activities such as: creation of a collaborative editing session, editing shared code, and other topics related to shared project artifacts. Also, each user's profile news feed shows their login history over a specified time period that is also illustrated by a graph to help predict their availability.

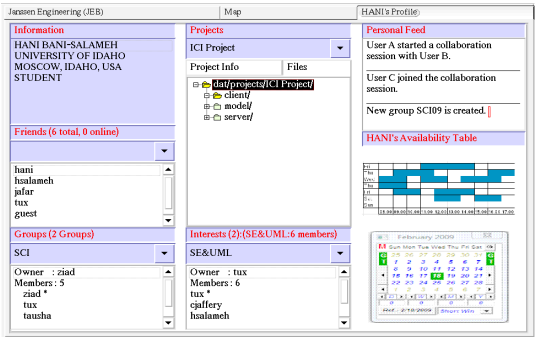

Figure 3. User's Profile

4.2. Virtual Environment and Project Presence

Developers create projects with significant permission differences. For "Group Open" projects, if a prospective new developer is one of the owner's friends (in the social network), they can join the project group at will. "Group Closed" projects require friend status along with permission to join the project. "Community Closed" projects require permission for joining the project. "Community Open" allow any developer to join the project. When a developer creates a project, they can set the project to public or private, and control access to its files. All the project files are stored in the server. The collaboration server is layered on top of an ordinary revision control server that remembers changes made to the project files. SCI exploits awareness of other team members; its text editor graphically depicts (using distinct background colors) the text lines with (a) pending updates to files that others have committed, (b) lines having uncommitted changes in other developers' copies of the files, and (c) lines that were recently viewed by others. This level of detail minimizes conflicts from concurrent

revision of the same code, and helps team members know when consultation or a meeting is in order.

Every project is allocated a space inside the CVE virtual environment when the project is created. Project spaces vary depending on their size; a project starts with a simple room sized for its initial membership. As the group becomes bigger the size and number of the project's rooms increase. Project spaces in the 3D environment can be laid out in different ways. If the project is not related to any other virtual spaces its creator can just switch to the 2D map tab, navigate the map and click to reserve an available space, and set up his room based on a specific coordinates. In many other cases, users can choose to lay out their projects' spaces next to the other rooms that they own or to which they are related, in order to make moving between the rooms easier. For larger projects and closely related projects, their spaces will form multi-room complexes that are connected in the virtual world. Also, users may choose to lay out their projects' spaces next to those projects, friends, or community members they share interest with or wish to ask assistance from.

Annetta and Holmes [3] define avatars as "graphical embodiments that convey a developer's identity, presence, location, and activity of others. Any user who joins a group is teleported to the group room, and each time the user opens the project within the CVE his/her avatar is teleported there. Since users typically are members of multiple projects, a teleport menu lists the rooms for each of their projects and interest groups. The presence of the avatars in a virtual room helps the developers feel the team's presence and encourages interaction between them across locations and cultures. Users can create rooms for specific groups and purposes. Each room's owner can control the invite, join, and access of the other members.

4.3. Activity and Group Awareness

Software development always presents coordination, communication, and collaboration problems, especially when teams are geographically distributed [4,5,6]. As a result, occasionally team members duplicate work, overwrite changes, or write code that affects other developers' code. This is due to the lack of awareness about what is happening in other parts of the project, and the other team members' activities.

Researchers have found a number of problems that occur in software development projects. It is difficult to: determine when developers make changes to the same code; communicate across time zones and different work schedules; find developers for closer collaboration or assistance; determine the developers who have the best knowledge of specific project artifacts. As Herbsleb and Grinter [7] state, "the inability to share at the same environment and to see what is happening at the other site" is one of the major factors in these difficulties. When working on geographically distributed teams, awareness of other team members' activities provides information that is important to build an effective collaboration. This awareness includes observing who is working with you, and their activities or plans. Developers can use knowledge of other team members' activities for many purposes that help the overall cohesion of the project. For example, knowing specific files and objects that another developer has been working on can give an indication of their expertise; tracking who made changes recently to a file suggests whom to ask before doing further changes; and information about who is currently active helps developers to find assistance and collaboration [8]. A few systems do track and visualize awareness information (such as Plantír [9], and TUKAN [10]).

To promote group awareness, SCI shows a tree of available users that provides awareness of one's team members. Developers can view others' status and profile, and check what project they are working on. Each team member is represented by a node in the tree and users can tell who is online and working in the SCI environment at a glance. Clicking on the user icon, developers can reveal further details about developers' activities; such as: what files they are currently editing or debugging, their active projects, their interest groups, and so on.

Users in the virtual environment, whose avatars reside in the project/group virtual room can reveal information about other users' tasks by hovering over their avatar's head and a tooltip or window will appear with a list of the files he accessed in the project decorated with different colors showing the files they are modifying at the moment, files they accessed in the last few days, and what new files they added to the project (see Figure 4). From the users/groups tree, users can start a variety of interactions, including text chat, VoIP session, and screen sharing (inviting others for pair programming, debugging, or code reviews). History of the source file changes can be saved as footnote (marker) beside the file tree node in the classes tree tab; another way is to add a marker before the first line (in the margin next to the code) of each changed method/procedure in the file. By clicking on this marker a list of the changes; their dates, the developers who made the changes and other details will appear.

In addition to team awareness, SCI provides artifact resource awareness by extending the "class browser". Files in the browser are provided with colored icons, indicating that someone is modifying a file at a particular moment, a file is been modified or accessed for the last few days, or never been accessed. By clicking a file node a small window appears with specific details; details such as who last changed/accessed the file and when.

Developers using repositories such as CVS [11] can normally see others' committed changes; however, their local uncommitted changes are not recorded in the shared repository. In addition to the ability to observe changes to the files during the collaborative editing sessions by

watching others editing, SCI records each user's uncommitted changes in the shared copy of the code. Awareness of uncommitted changes helps avoid conflicts. Notations in the left margin of the code inform the user where team members have made changes.

Figure 4. Project Files Awareness shows files a user is working on (red), accessed recently (green), and added recently (blue).

SCI provides session awareness, a tree structure of the available session. By hovering over a session icon (node), a tooltip will appear with the session history; showing the owner of the session, who is editing the file at a particular moment, and a list of all the current members and the members who left the session; each decorated with a different icon. Awareness information of the currently edited files and who is currently editing or debugging a specific file helps simplify the creation of collaborative IDE editing/debugging sessions. Developers that observe others working in the same file that they are editing or need some assistance while working on it, can do an invitation and start a collaborative session.

A log file on the server in each project directory stores the login/logout activity for each member of the project. This history is available to from inside the environment by clicking on a button in the project environment, helping users predict other developers' availability.

To summarize, SCI provides the developer with activity awareness information that covers the following categories: social awareness (the presence of one's collaborators and community members), action awareness (awareness of what collaborators are doing or what they have recently done), and artifacts and work items "files" awareness (information about all the different tasks and files or sub-projects that make up the overall project).

5. Related Work

In the past few years, there have been advances in the development and adoption of social awareness inside IDEs, which allow developers to collaborate and form online communities. Many open source projects use wikis (for example wiki.eclipse.org and wiki.apache.org) and some projects provide the developer community with chat capabilities. Several IDEs integrate collaboration and awareness similar to SCI. This section highlights various existing systems and research that provide interactive collaboration and awareness for multiple phases of program development, such as Eclipse (www.eclipse.org).

A notable working example of a collaborative IDE is Jazz, a research project at IBM that adds a set of collaboration and features for the Eclipse IDE [12] [13] [14]. The objective is to help developing the collaboration within the group. Jazz provides a facility similar to an IM buddy list to monitor who is online and whether they are coding or not. Developers can initiate chats, or use different communication methods such as screen sharing and VoIP telephony. Jazz also provides some awareness features to provide the developer with some awareness features of the activities of other team members [13].

Stellation (www.eclipse.org/stellation), is another Eclipse open source effort led by IBM Research that supports the notion of activities and simplifies collaboration and provides awareness of team members' changes. It enables developers to manage relevant work, notify the team of their current work, be informed of changes pertaining to their own activities, and provides a context for persistent conversations.

Palantir [14], another Eclipse plug-in supports awareness features by showing which artifacts have been changed by which developers and by how much. Palantir provides workspace awareness such that developers can monitor other teams' activities while working on their current task and without the need for switching contexts.

CollabVS [15] allows developers to work together ad-hoc. Tools provided by CollabVS include IM, audio and video communication, programming, testing, and debugging. CollabVS provides: 1) real-time presence information to let users know what other team members are doing (what users are online and whether they are editing, debugging, or engaged in an instant messaging session); 2) contextual presence information facilitates finding relevant information and people quickly.

To our knowledge, no existing tool does the full "social networked IDE" features of SCI. Jazz, the closest relative, supports parts of social networking. Like SCI, Jazz increases users' awareness of people, resources, and activities, and fosters communication among team members. Jazz and SCI support synchronous chat discussions. Jazz's team-centric discussion boards are similar to SCI's asynchronous news feeds. User profiles are not supported by Jazz. Jazz gives awareness of the committed code changes in the code repository, but SCI also provides the developer awareness information of uncommitted code changes, the currently edited files.

The major difference between SCI and almost all the related work is the integration of social network features inside the software development environment. Most cited

projects have some social network-like features, but not others. Open source platforms such as SourceForge [16], provide simple awareness mechanisms along with configuration management functionality. It is difficult to maintain awareness across SourceForge's many communication channels. Developers are informed mainly of bugs and conflicts related to specific project artifacts and not the other developer activities [9]. In contrast, SCI integrates multiple social network information sources and provides the user with complete awareness about the other developers and the project artifacts. It provides what most other open source projects are missing: the overall view of other developer's workspace activities.

6. Conclusions and Future Work

There is great potential in exploring tool support for the social side of software development. Collaboration plays a crucial role in software development. For this reason, continuing to improve the collaborative tools available inside integrated development environments is of great potential benefit. Collaborative tools can be used alongside a non-collaborative IDE, but integration adds qualitative and quantitative awareness information and reduces the cost of collaboration during the development process, particularly for distributed teams.

SCI is now composed of presence and activity awareness information tools, integrated development environment, and collaboration tools that reside within a virtual collaborative development environment. The merger of these tools makes them of more benefit to the development community. The IDE benefits from the provided awareness support and communication context.

This paper described the early stages of a project to develop and evaluate a social real-time collaborative IDE to make communication, and collaboration more productive in software development teams. We aim to make it as complete and efficient as possible for the software development community. News about the project will be available at cve.sf.net/socialide.

The tool presented in this paper will test two hypotheses. One hypothesis is that integrating social tools inside an IDE helps developers save time lost switching between different tools. The second hypothesis is that the combination with a virtual environment will enrich the collaborative IDE with the social activities. The IDE will benefit from the awareness support and communication context provided by the virtual environment. The virtual environment provides a "social presence", the sense in which developers feel that others are present in the collaborative development environment. Future work will evaluate the correctness of those hypotheses.

7. References

[1] K. Ehrlich, G. Valetto, and M. Helander, Seeing inside: Using social network analysis to understand patterns of collaboration and coordination in global software teams, pp.297-298, International Conference on Global Software Engineering (ICGSE 2007), 2007.

[2] E. Prasolova-Førland and M. Divitini, Collaborative virtual environments for supporting learning communities: an experience of use, Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work, Sanibel Island, Florida, USA, 2003, pp. 58 – 67.

[3] L. A. Annetta, S. Holmes. Creating Presence and Community in a Synchronous Virtual Learning Environment Using Avatars. International journal of instructional technology and distance learning, 2006. pp. 27-43.

[4] P. Bouillon, J. Krinke, and S. Lukosch, Software Engineering Projects in Distant Teaching, 18th Conference on Software Engineering Education & Training (CSEET'05), 2005, pp.147-154.

[5] C. Gutwin, R. Penner, and K. Schneider, Group Awareness in Distributed Software Development, Proceedings of the 2004 ACM conference on Computer supported cooperative work, Chicago, Illinois, USA , 2004, pp. 72 – 81.

[6] R. E. Grinter, J. D. Herbsleb, and D. E. Perry, The Geography of Coordination: Dealing with Distance in R&D Work, Proceedings of the 1999 International ACM SIGGROUP Conference on Supporting Group Work, Phoenix, Arizona, USA, 1999, pp. 306-315.

[7] J. D. Herbsleb and R. E. Grinter, Architectures, Coordination, and Distance: Conway's Law and Beyond, IEEE Software, vol. 16, no. 5, Sep./Oct, 1999, pp. 63-70.

[8] K. Schneider, C. Gutwin, R. Penner, and D. Paquette, Mining a Software Developer's Local Interaction History, Proceedings of the International Workshop on Mining Software Repositories, Saint Louis, Missouri, USA, 2005.

[9] A. Sarma, Z. Noroozi, and A. der Hoek, Palantír: Raising Awareness among Configuration Management Workspaces, Proc. of the ICSE 25, Portland, Oregon, 2003, pp. 444–454.

[10] T. Schümmer, Lost and Found in Software Space, Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Vol. 9, 2001.

[11] CVS. Concurrent Version System. Available online at http://www.nongnu.org/cvs.

[12] L. Cheng, C. de Souza, S. Hupfer, J. Patterson, and S. Ross, Building Collaboration into IDEs. ACM Queue 1, 9(2003-2004), pp. 40-50.

[13] L. Cheng, S. Hupfer, S. Ross, J. Patterson, B. Clark, and C. de Souza, Jazz: a collaborative application development environment, Demonstration at OOPSLA 18, Anaheim, CA, USA, pp. 102-103.

[14] R. Ripley, A. Sarma, and A. van der Hoek, Workspace Awareness in Application Development, Proceedings of Eclipse Technology eXchange Workshop, Vancouver, Canada, 2004, pp. 17–21..

[15] Collaborative Development Environment using Visual Studio. Available at http://research.microsoft.com/en-us/projects/collabvs/default.aspx.

[16] SourceForge.net. available at http://sourceforge.net/