A Social Collaborative Virtual Environment for Software Development

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 often require information and expert help that are not supported by conventional development environments. This paper presents a social development environment called SCI (Social Collaborative IDE) that unifies the concepts of social network and collaborative development environment. SCI integrates presence and activity awareness information and collaborative program development tools. The activity awareness information provides inter-project and inter-session sharing and awareness of other developers' behavior. The collaboration tools provide a wide range of facilities for synchronous and asynchronous collaboration and information sharing between team members. SCI presents a framework that manages pending invitations of team members.

KEYWORDS: Collaborative Development Environment, Social Networks, Software Development, Non-Player Characters, SCI.

1. INTRODUCTION

Software engineering is a team effort. Many projects involve a broad community of individuals from different institutions [1]. Software development is a very knowledge intensive and social process where developers exchange information and code fragments, connect with others who share technical interests, post messages in the team or community weblogs, and answer questions and discuss issues in public mailing lists [2].

Collaborative Development Environments (CDEs) are tools where developers meet to solve design and coding problems, and create software products. During these activities, developers often need to interact with like-minded individuals who have expertise in particular technical problem domains. The main goal of a CDE is to support the development process during the whole life cycle of the project [3].

Social networks build online communities of people with common interests or activities. Our hypothesis is that CDEs and social networks complement each other in collaborative software development, because software development communities *are* social networks. This leads to a new category of tool that falls under the CSCW umbrella, the Social Development Environment (SDE). SCI (Social Collaborative IDE) is a tool in this category that integrates the development environment and the social network. This paper presents the design of SCI and describes the interaction between its social networking feature and work collaboration tools.

In order to maximize their productivity, development team members 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. They need to be aware of the other team members' primary activity patterns and project contributions [4, 5].

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

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 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. The remainder of this section describes the two foundations for the design of SCI.

1.1. Social Software Development

In this paper the term *social software development* refers to software development in collaborative communities with social relationships. Software development requires interaction between the people involved in the development process. For this reason, social activities form a big part of their daily work.

In software development, developer networks are an instance of object-centered sociality [6], where developers do not usually interact merely to socialize, as in conventional social networks. In contrast, they interact and collaborate through shared project artifacts.

1.2. Collaborative Development Environments

"While traditional integrated development environments focus on improving the efficiencies of individual developers, collaborative development environments focus on improving the efficiencies of the entire development team [7]."

Booch and Brown [3] define the collaborative development environment as "a virtual space wherein all stakeholders of a project – even if distributed by time or distance – may negotiate, brainstorm, discuss, share knowledge, and generally labor together to carry out some task, most often to create an executable deliverable and its supporting artifacts". Also they present five different stages of CDEs' maturity, where each relies on the previous one. Those stages are artifact storage support, collaboration mechanisms support, advanced artifact management, assisting team management, and finally "asset and information services that encourage a vibrant community of practice".

This paper focuses on enhancing the collaborative development environment with community support and presence information that will encourage collaboration during the development process.

The rest of the paper is organized as follows. Section 2 introduces SCI. Section 3 lists the awareness requirements. Section 4 presents some technical challenges. The design, architecture components, and presence features are described in Section 5. Section 6 presents related tools. Section 7 presents the future work and concludes the paper.

2. SCI OVERVIEW

As mentioned earlier, SCI extends the ICI development environment. ICI in turn extends the CVE virtual environment with communication and collaborative features (see Figure 1). The inner oval represents the CVE collaborative virtual environment, where users interact within a 3D virtual world. CVE provides developers with a general real-time 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. In the middle oval, ICI provides developers with synchronous tools to collaborate in solving their programming problems. SCI is the outer oval, which provides developers with presence and awareness information for their groups, users, projects, and sessions. SCI's asynchronous features help users to select and coordinate their active synchronous collaborations.

In general, asynchronous tools drive the use of the synchronous tools, and the two categories complement each other. CVE, ICI, and SCI are complementary tools that work together to provide a unique development environment.

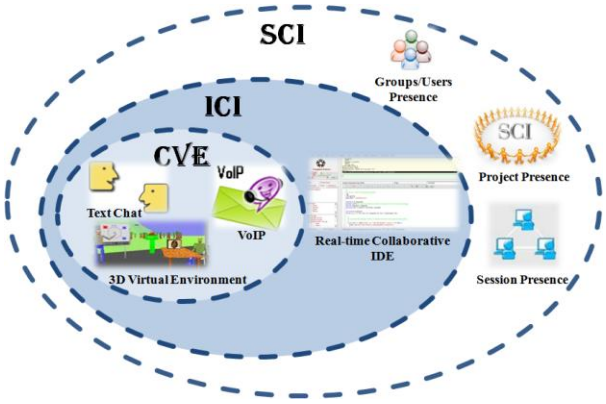

Figure 1. The SCI Architecture

3. AWARENESS REQUIREMENTS

Distributed developers need to maintain general awareness of the entire team, as well as detailed awareness of people of special interest, such as people that a developer is working with, or from whom the developer wishes to ask assistance. Developers in distributed software development projects maintain their awareness primarily through text-based communication tools, such as: mailing lists, email, and chat systems.
Developers benefit from awareness of what parts of the code others are working on, their plans, and what their areas of expertise are, as well as who has joined (or left)

the team. They gain this information from different sources, such as: (1) watching the team/group news feed; (2) observing the people who access the project artifacts; and (3) observing changes to the project artifacts committed to shared file systems or revision control repository. This information helps developers to be up-to-date about both changes to the project artifacts and the activities of other distributed team members.

Also, mailing lists and news feed help developers gather information about others when they wish, for example, to find out who are the experts in an area. They can gather information about who are the experts in a specific part of the code, by checking the change history of the code and the time each team member spent working on this code.

4. Technical Challenges

Like any other distributed and groupware system, SCI has many technical challenges that need to be addressed. This section touches on a handful of these challenges, some of which SCI addresses and mitigates, while others remain open issues.

The interactions between distributed team members are disadvantaged due to the lack of the rich social communication and coordination that is only possible when team members are collocated. Being aware of these challenges is important for evaluating the effectiveness of a groupware system. The following are some of the problems that are facing distributed teams [8]:

1. Distributed teams lose awareness of social interactions and other members' activities. They are limited in the number and type of spontaneous interactions that occur with other team members.
2. Team members cannot easily observe what other team members are working on, the status of their activity, and what tasks are underway.
3. In distributed environments, communication relies on lower-bandwidth tools and applications such as: phone and email. Also, they are constrained by transaction delays in comparison with face to face interactions and encounters.
4. In order for the distributed teams to manage their project artifacts, they face numerous challenges, including version control and user access management.

5. SOLUTION DESIGN

SCI's goal is to provide easy access to as much of the developers' community information as possible, and make collaboration and communication more tangible and visible within the community. As teams become bigger, the social challenges of managing and coordinating the team's work assume a greater impact in the overall success of the project.

SCI is composed of two parts: 1) an information part that provides presence and activity awareness information about each team member, their activity in the project, tasks assigned to each member, and progress on these parts of the project; 2) a tools part, consisting of tools to facilitate communication, collaboration, and information sharing between team members.

5.1. SCI Interface Components

SCI's major components (Figure 2) are:

- Collaboration Spaces (A).
- The ChangeBar (B) uses color coding to depict the user who committed the most recent updates or changes to each line.
- Activities on the current collaboration space (C).
- Chatting (Text and VoIP) (D).
- Tabs(E) : News Feed Tab, and Activity Tab (that shows both of F and G).
- Session Tree (F).
- Mini Tab Set (G): includes Users Tree, Groups Tree, and Projects Tree.

The social parts (F) and (G) represent part of the awareness information that users get "for free" while concentrating on their project tasks.

Tabs (E), show social awareness of the users, their status (online, offline, or idle), active collaborative sessions and members of each session. Information in the tabs show the presence of the available teams (groups), the virtual rooms, and who belongs to each team. Also, they show 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. More detailed information about the activity awareness is shown in sections 5.2 and 5.3.

In addition to the components represented in Figure 2, SCI presents the following components:

- ***Special Interest Groups:*** Developers can join special interest groups (SIGs) where they can find other team members or developers that share the same interest. The system began with four original SIGs: Java Group, C++ Group, Unicon Group, and Software Engineering Group. A smart NPC (Non-Player Character) hosts each SIG's virtual room. The NPC is available there to offer help and give materials related to those specific subjects. Construction of the NPCs'

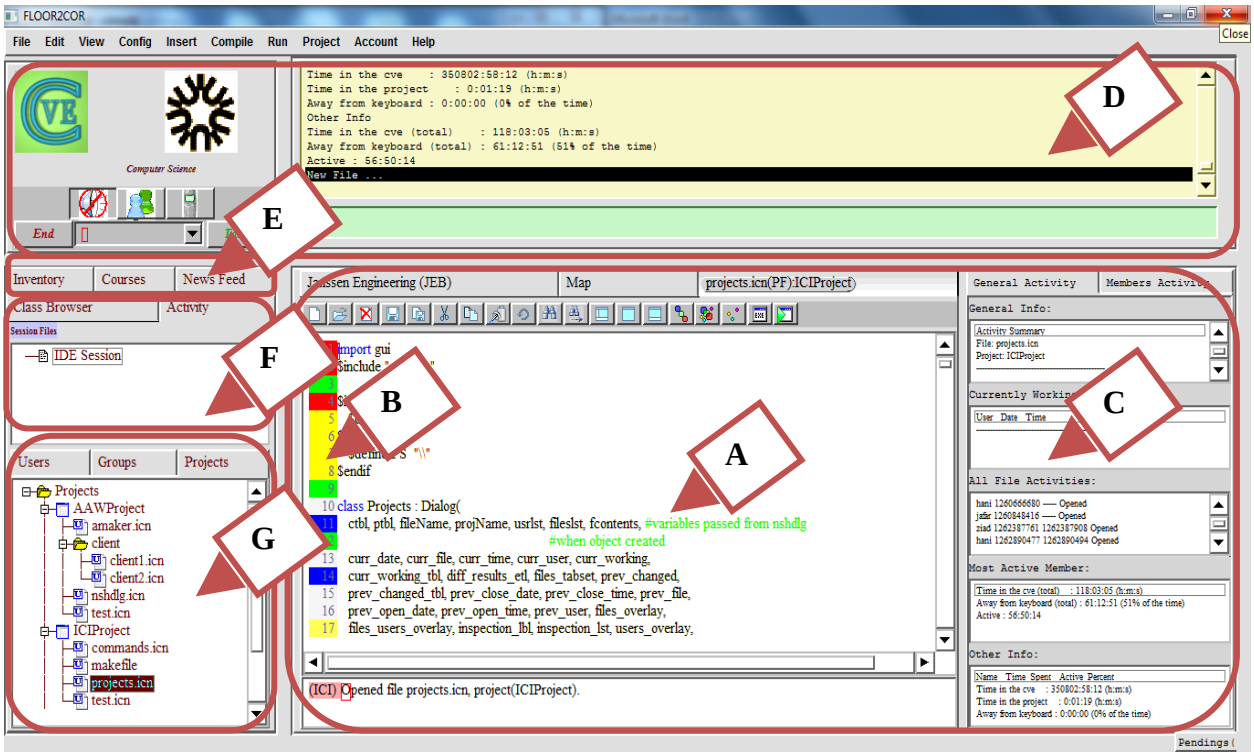

Figure 2. A View of the SCI Development Environment

expert system rules for interaction with users is a collaborative activity of the SIG members. Users can interact with the NPC when there is nobody else available to help.

- ***Profiles and News Feed:*** Users are allowed to view others' information within the community circle (teams or groups). Also, they can view information about the community and the project friends circle using the available news feed, and mini-feed that are available in every user profile (see Figure 3).

5.2. The Effect of 3D Virtual Environment on Project Presence

Developers can create different types of 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), then they obtain permission to join the project group automatically. "Group Closed" projects require friend status along with a request for permission in order to join the project. "Community Closed" projects require permission for joining the project regardless of whether the second developer is a friend or not. "Community Open" projects allow any developer to join the project.

All the project files are stored in the collaboration server, layered on top of an ordinary revision control server that remembers all 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. Any user who joins the group causes their avatar to be teleported to the group room, and each time the user opens the project within the CVE his/her avatar is teleported there. Being aware of the fact that users typically will be members of multiple projects, users will have a teleport menu (list) with the rooms for each of their projects and interest groups. It is hoped that the presence of the avatars in a virtual room gives the developers the feeling of the team's presence and encourages interaction between them regardless of the variety of locations and cultures. Users can create rooms for specific groups and purposes. Each

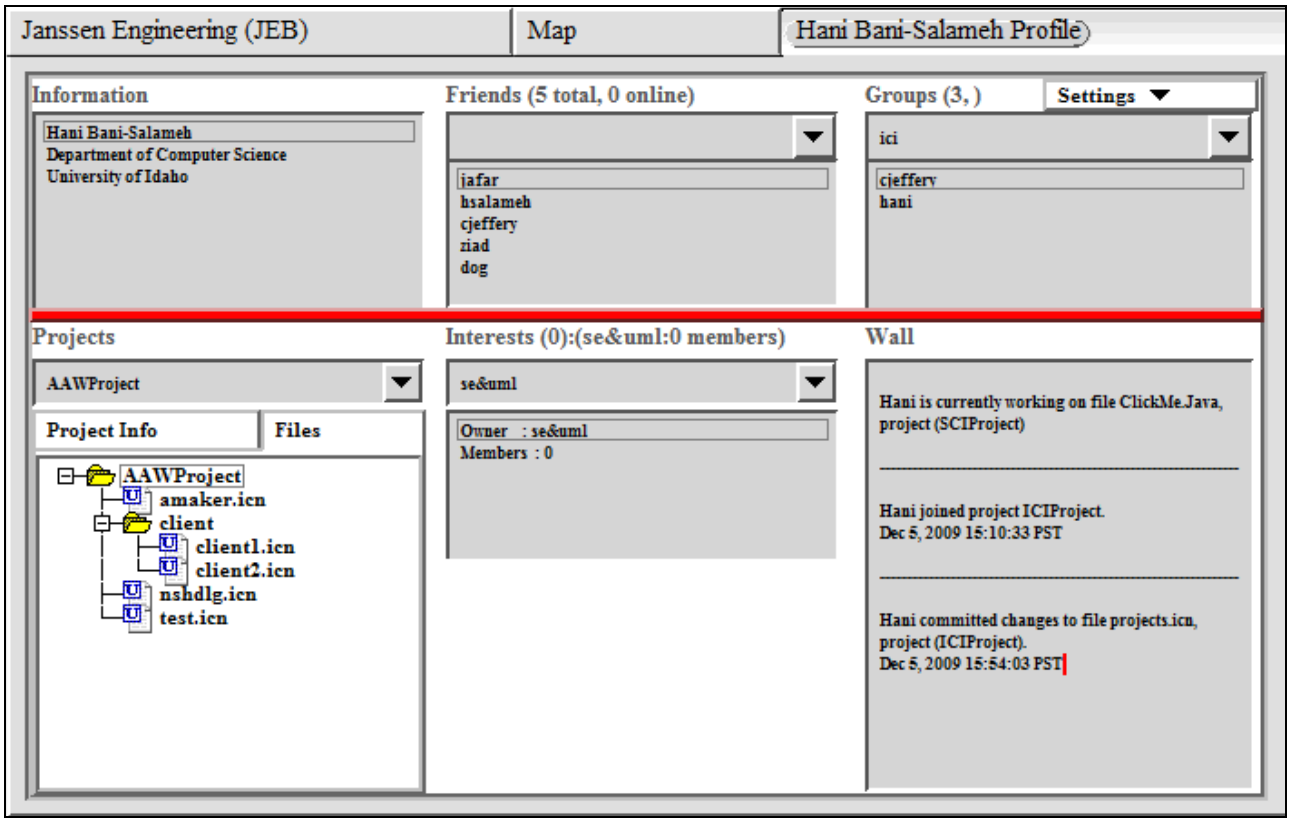

Figure 3. User's Profile

room's owner can control the invite, join, and access of the other members.

5.3. Activity and Group Awareness

There are often situations in distributed software development where team members duplicate work, overwrite changes, or write code that breaks other developers' work in progress.

Researchers have found that it is difficult to: determine when developers are making changes to the same piece of the project; communicate with others due to time zones barriers and different work schedules; find developers for closer collaboration or assistance; determine the expert/right developers who have the knowledge about the project different artifacts. As Herbsleb and Grinter [9] state, lack of awareness- "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 with groups who are geographically distributed, 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 what are their activities or plans. Developers can use the knowledge of other team members' activities for many other purposes that help to assist the overall cohesion of the project. For example, knowing the specific files and objects that another developer has been working on can give a good indication of their expertise; tracking who made changes most recently to a particular file gives an indication of whom to ask before doing further changes; and gathering information about who is currently active can help developers to find more timely assistance and possible real-time collaboration [10].

To promote group awareness, SCI supports a tree structure of available users organized by project and interest group membership 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. Figure 4 is showing the files the users are currently working on (red), files they accessed in the last few days (green), and what new files they added to the project (blue) (see Figure 4).

Figure 4. Project Files Awareness

From the users/groups tree tab, 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.

Developers who use version control systems (such as SVN (http://subversion.tigris.org), or CVS (http://www.nongnu.org/cvs) can observe/check others' changes by checking the committed changes history into the software repository; however, their local uncommitted changes with the project artifacts 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, so changed uncommitted pieces of the code appear highlighted with a different color. The benefit from saving/recording local uncommitted changes is to provide the system with information that can be used to support awareness in distributed software development and to assist them in avoiding making changes that overlap and conflict with others' changes.

While working on a project file, developers are alerted during their editing or debugging session of changes to the same file: who is doing those changes, what portion of the code are affected, and whether they have committed those changes or not. When they open a file for editing or debugging, they get a notice about others currently editing the same file. Developers can commit changes in the files they finish editing. Once they decide to commit the changes, a window will popup showing who is currently editing the same file, and what time they started, also they can view who previously changed the file, when they started, and when they finished. They can also compare their own version of the file with the previously saved copies after each commit a developer made. The supported awareness features introduced here can help team members coordinate their work and avoid conflicts and duplication of effort regardless of their locations. SCI provides session awareness, a tree structure of the available session.

By hovering over a session icon (node), a tooltip appears with the session history; showing information about who is editing the file at a particular moment, and a list of all the current members and the members who left the session. Awareness of who is currently editing or debugging a specific file helps simplify the creation of collaborative editing and 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.

There is a log file saved in the server, in each project directory, to store the login/logout activity for each member. This logging history is available to the user to view easily, and help 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).

5.4. Pending Invitation/Request Management

As mentioned earlier, collaborative software development revolves around interactions between developers, including file and project sharing invitations, and editing and debugging session invitations. Social features add different kinds of interactions such as friend requests, group invitations, and project discussions. In such an environment, users get a lot of invitation traffic. This creates a need for a framework that can transparently manage all these kinds of invitations.

Managing invitations and requests is required for two main reasons. First, the recipient of the invitation may be unavailable. In this case, invitations can be held until the invited user gets a chance to see them. The second reason is that managing invitations is required from a usability and scalability point of view. For example, an invitation to a collaboration session might intuitively be delivered via a popup dialog, but such intrusions do not scale well for busy users. Managing several opened windows and switching between them can be cumbersome especially if there is a large number of incoming invitations.

The primary emphasis in the design of the proposed system is to minimize the transition effort between individual work, to collaborative session, interaction with other users, and then back to individual work, so that this can easily occur dozens of times during the course of a work period. A Facebook-style visible indicator of pending invitations is implemented adding the ability to review the queue at one's convenience to see who has been waiting for what, and for how long.

Once a user gets an invitation, they can choose one of several provided actions. They can accept, reject, or discuss the invitation/request by starting a chat session or write an email to reply to the invitation/request. In some cases such as collaborative IDE editing and debugging sessions, users may forward the invitation to another user or expert. For example, user "A" invited user "B" for a collaborative IDE session, and user "B" is busy assisting another team member. In this case, user "B" might forward the session to user "C", and the user will make an accept or reject action. All invitations/requests are placed in the pending list until an action is taken, except the initiation and joining of the collaborative IDE sessions. The invitation is removed from the list when either the sender or the receiver of the invitation signs out of the system for any reason. The system provides a prioritization for invitations. Invitations from friends and project team members get higher priority than other invitations and requests. Figure 5 shows a preliminary implementation for the pending invitations/requests management window. Current user pending invitations'/requests' are shown in the main area at left (A). Users can select the action to reply to the pending item from (B). Information about the selected pending item appears in (C).

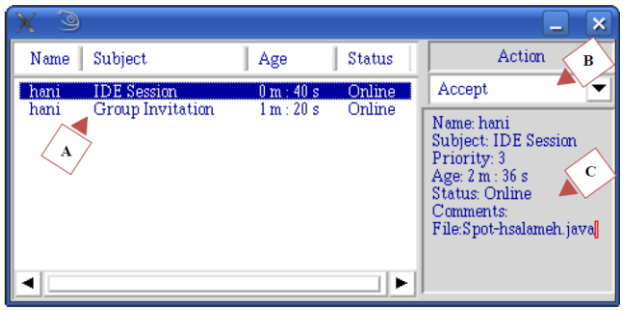

Figure 5. Invitation/Request Management

6. RELATED WORK

Several IDEs integrate collaboration and awareness features. This section highlights various existing systems that provide interactive collaboration and awareness for multiple phases of program development.

Eclipse [11] is an open source development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle [12]. Eclipse itself does not support code-level collaboration, but a new Eclipse Communication Framework project aims to support the development of distributed Eclipse-based tools and applications and to allow the Eclipse code repository and project model to be shared and collaboratively edited. Eclipse provides an API to perform basic sharing, along with some prototype client applications.

A number of other Eclipse-based projects focus on the integration of collaborative features into IDEs. GILD is an example of a project that provides cognitive support for novice programmers and support for instructor activities [13].

CodeBeamer is a commercial product that has plug-ins for integrating collaborative capabilities into IDEs such as chatting, messaging, project management, and shared data [14]. Another example is Sangam, a plug-in for the Eclipse platform that features a shared editor and chat for pair programming [15].

Stellation (http://www.eclipse.org/stellation) is an open source effort (led by IBM Research) that introduces a fine-grained source control that supports the notion of activities and aims to simplify collaboration and provide awareness of changes to team members [16]. 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.

Palant´ır [17, 18] is another Eclipse plug-in that supports awareness features by showing which artifacts have been changed by which developers and by how much. Palant´ır provides workspace awareness such that developers can monitor other teams' activities while working on their current task and without the need for switching contexts.

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 [16, 19]. It aims to develop collaboration within the group. Jazz provides a facility similar to an IM buddy list to monitor users' status and whether they are coding or not. Developers can chat, or use different communication methods such as screen sharing and VoIP telephony. Jazz also provides some awareness features to make developers aware of the activities of other team members [11]. There are other Eclipse plug-in that support awareness features of project artifacts and other teams' activities such as Stellation (http://www.eclipse.org/stellation), and Palantir [18].

Jazz, the closest relative, supports many precursors to social networking. Like SCI, Jazz focuses on increasing the user's awareness of people, resources, and activities, and on fostering communication among team members. Both Jazz and SCI support synchronous chat discussions. Also, Jazz provides team-centric discussion boards that are comparable to the asynchronous news feed supported by SCI. User profiles are not supported by Jazz; in SCI developers can view other developer's profile and see their friends, groups, projects, and activities. Jazz supports awareness of the committed code changes with respect to the code repository. In contrast, SCI provides awareness information of the committed and uncommitted code changes, of the currently edited files, and indicates who is responsible for the changes.

CollabVS [20] allows developers to work together whether intentional or ad-hoc. For example, pair of developers can agree to work together at a scheduled time and work together using CollabVS. CollabVS allows for opportunistic collaboration as well. Developers can carry out various activities using the tools provided by CollabVS, including IM, audio and video communication, in addition to the standard activities such as: programming, testing, and debugging. Also, CollabVS provides two kinds of presence 1) real-time presence that make the user aware of what other team members are currently doing (it shows what users are online and whether they are editing, debugging, engaged in an instant messaging session, etc); 2) contextual presence facilitates finding relevant information and people quickly.

Collab.net [3] is a commercial CDE that has both a public and a private face. Collab.net's public face is SourceForge, an open source CDE that focuses on the development of open source software. SourceForge serves as a host to approximately 230,000 projects such as CVE (http://cve.sf.net). Collab.net's private face is SourceCast (http://sourcecast.org/) a CDE that supports a number of important features not in SourceForge, such as greater security that is not a big issue for open source development.

The CDE tools presented previously in this section are categorized in Table 1. Table 1 shows that CDE tools vary in the number of supported features; most CDE tools are designed for specific purposes and in general do not need to support all software development tasks.

In the design of the SCI framework, a major goal is to provide many CDE features for application developers to utilize. However, instead of writing tools for all purposes of SCI, the research presented in this paper provides a framework that supports key categories listed in the features table, while focusing on the awareness, social presence, and social network features, and mainly the 3D virtual environment integration with the gains from the interaction among users' avatars and their online presence.

7. CONCLUSION AND FUTURE WORK

Social support for software development is an important emerging field of research. Conventional single-user tools do not provide the needed environment for smooth collaboration between distributed developers due to the size and complexity of today's development projects. CDE tools that support and provide project artifacts' updates in real time have the potential to raise the level of communication, and coordination between distributed developers.

Most current CDEs have inherent limitations, including little support for awareness and online presence, missing social networking features, and weak support for source code repositories' features. The multitude of tools increases the friction that results from switching among different tools.

Table 1. Features of Existing CDE Systems

		GILD	CodeBeamer	Stellation	Palant'ir	Jazz	JBuilder 2008	CollabVS	SCI
	Synchronous Features	_	Partial	Partial	Partial	X	X	X	X
	Asynchronous Features	_	Partial	Partial	Partial	X	_	X	X
	Artifact Management	X	_			X	X	_	X
	Purpose	EDU	COM	DEV	DEV	COM	GEN	_	GEN
	Multiple Language Support	_	_	_	_	X	_	_	X
	3D Virtual Environment	_	_	_	_	_	_	_	X
Awareness	Social/Presence	_	_		_	X	_	X	X
Awareness	Historical	_	_	X	_	X	_	X	X
Awareness	Group-Structural	_	_	X	_	X	_	X	X
Awareness	Workspace (Artifacts)	_	_	X	_	X	_	X	X
SN Features	Groups	_	_	_	_	X	_	_	X
SN Features	Friends	_	_	_	_	X	_	_	X
SN Features	Help/Support	_	_	_	_	X	_	_	X
	Purpose: EDU (Education), COM(Commercial), DEV(Development), GEN(General)								

This paper presents a social collaborative development environment that addresses these problems and eliminates several current limitations. SCI is composed of a presence and activity awareness information component, an integrated development environment, and collaboration tools that all reside within a single environment, which serves as a virtual collaborative development environment. The merger of these tools increases their benefit to the development community.

No existing tool has yet blended full "social networked IDE" features with a 3D virtual environment as found in SCI. The major difference between SCI and almost all the related work cited in this paper is the integration of social network features inside the software development environment. Most of the cited projects provide some social network-like features, but not others such as user profiles and news feed. Open source development, such as SourceForge [21], provide simple awareness mechanisms along with configuration management (CM) functionality. SourceForge has limitations in terms of what information is shared, when is shared, and how is presented to the developers. It is difficult to maintain awareness across SourceForge multiple communication channels. Perhaps the limitation appears where most open source projects inform developers only of conflicts related to specific project artifacts and ignore the other developer activities [22]. 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.

SCI supports development in mainstream languages, such as C, C++, and Java. Depending on the expected research results, augmenting related and specific awareness information, online presence, and social network features within a single environment makes SCI a welcomed development environment between developers and teams working in a distributed environment.

We are in the later stages of a project to develop and evaluate a social real-time collaborative IDE that will help the software development teams and make communication, and collaboration effective and more productive. We aim to make it as complete and efficient as possible and attractive to the software development community. News about the project will be available at http://cve.sf.net/socialide.

The research project presented in this paper plans to test two hypotheses. The first hypothesis is that integrating social tools/features inside an IDE will help the developers save the time required for 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. Regardless of the fact that virtual environment represents a collaborative social environment and could provide social bonds between developers. It could provide a "social presence"- the sense in which developers feel that there are others present in the collaborative development environment. Future work will include case studies to proof the correctness of those hypotheses.

ACKNOWLEDGMENT

This work was supported by a grant from the National Science Foundation (NSF-ATE); grant agreement number DUE-0402572.

This work was supported in part by the Specialized Information Services Division of the U.S. National Library of Medicine.

REFERENCES

[1] K. Ehrlich, G. Valetto, and M. Helander, "See-ing 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] P. N. Robillard, "The role of knowledge management in software development." In Communications of the ACM, 42(1):87-94, 1999.

[3] G. Booch and A. Brown, "Collaborative development environments." Advances in Computers, 59:2–29, 2003.

[4] E. Prasolova-Førland and M. "Divitini, Colla-borative 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.

[5] H. Bani-Salameh, C. Jeffery, and J. Al-Gharaibeh, "SCI: Towards a Social Collaborative Integrated Development Environment." Workshop on Social Computing in Education (WSCE09) that held in association with the 2009 International Conference on Social Computing (SocialCom), Vancouver, Canada, August 2009.

[6] W. Reinhardt and S. Rinne, "An Architecture to Support Learning, Awareness, and Transparency in Social Software Engineering." Proceedings of the Special Track on MashUps for Learning at ICL 2009, 2009.

[7] G. Booch and A. Brown, "Collaborative Development Environments." Jan 11, 2007. Available at http://www.ddj.com/architect/196900222. Accessed February 15, 2010.

[8] W. Pole, "Virtual Workgroups: Hidden Challenges to supporting Distributed Teams." A whitepaper 1996-2005.

[9] 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.

[10] 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 (MSR), Saint Louis, Missouri, USA, 2005.

[11] "Eclipse Communication Framework." http://www.eclipse. org/ ecf/documentation.php.

[12] "Eclipse Platform Technical Overview." Object Technology International Incorporated, February 2003(updated for 2.1; originally published July 2001). Available at http://www.eclipse.org/ whitepapers/eclipse-overview.pdf. Accessed February 15, 2010.

[13] M.-A. Storey, J. Michaud, M. Mindel, et al., "Improving the Usability of Eclipse for Novice Programmers." OOPSLA Workshop: Eclipse Technology Exchange. pp. 35-39, Anaheim CA, October 2003. http://gild.cs.uvic.ca/docs/public ations/oopsla.pdf. Accessed February 15, 2010.

[14] CodeBeamer: http://www.intland.com and http://www. codebeamer.com . Accessed February 15, 2010.

[15] C. Ho, S. Raha, E. Gehringer, and L. Williams, "Sangam: A Distributed Pair Programming Plug-in for Eclipse," Eclipse Technology Exchange (Workshop) at the Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) 2004.

[16] 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.

[17] 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.

[18] L. Hattori and M. Lanza, "An Environment for Synchronous Software Development." In Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), Vancouver, Canada, May 16-24, 2009.

[19] L. Cheng, S. Hupfer, S. Ross, J. Patterson, B. Clark, and C. de Souza, "Jazz: a collaborative application development environment," Demonstration at the 18th annual ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications, Anaheim, CA, USA, pp. 102-103.

[20] "Collaborative Development Environment using Visual Studio." Available at http://research.microsoft.com/enus /projects/collabvs/default.aspx. Accessed February 15, 2010.

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

[22] A. Sarma, Z. Noroozi, and A. der Hoek, Palantír: "Raising Awareness among Configuration Management Workspaces," Proceedings of the Twenty-fifth International Conference on Software Engineering, Portland, Oregon, USA, 2003, pp. 444–454.